எளிய தமிழில் JavaScript

JS

து.நித்யா

எளிய தமிழில் JavaScript

JS

துர.நித்யா

□□□□ □□□□□□□

JavaScript

□□.□□□□□□

nithyadurai87@gmail.com

மின்னூல் வெளியீடு :

கணியம் அறக்கட்டளை

kaniyam.com

அட்டைப்படம், மின்னூலாக்கம் :

பிரசன்னா udpmprasanna@gmail.com

JavaScript என்பது இணையப் பக்கங்களை உருவாக்கும் ஒரு கணினி மொழி. இதை, இந்த நூல் எளிமையாக அறிமுகம் செய்கிறது.

தமிழில் கட்டற்ற மென்பொருட்கள் பற்றிய தகவல்களை “கணியம்” மின் மாத இதழ், 2012 முதல் வெளியிட்டு வருகிறது.இதில் வெளியான JavaScript பற்றிய கட்டுரைககளை இணைத்து ஒரு முழு புத்தகமாக வெளியிடுவதில் பெரு மகிழ்ச்சி கொள்கிறோம்.

உங்கள் கருத்துகளையும், பிழை திருத்தங்களையும் editor@kaniyam.com க்கு

மின்னஞ்சல் அனுப்பலாம்.

http://kaniyam.com/learn-javascript-in-tamil என்ற முகவரியில் இருந்து இந்த நூலை பதிவிறக்கம் செய்யலாம். உங்கள் கருத்துகளையும் இங்கே பகிரலாம்.

படித்து பயன் பெறவும், பிறருடன் பகிர்ந்து மகிழவும் வேண்டுகிறோம்.

கணியம் இதழை தொடர்ந்து வளர்க்கும் அனைத்து அன்பர்களுக்கும் எமது நன்றிகள்.

த.சீனிவாசன்

tshrinivasan@gmail.com

ஆசிரியர்
கணியம்

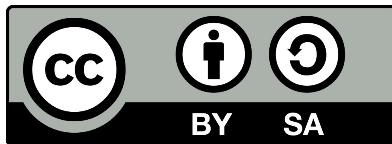
editor@kaniyam.com



நூல் மூலம் :

http://static.kaniyam.com/ebooks/Learn-Javascript-in-Tamil.odt




BY
SA

□□□□□□□□ □□□

அழகான, பல்வேறு வசதிகள் கொண்ட இணைய தளங்களை உருவாக்க HTML, CSS, JavaScript, Jquery ஆகிய நுட்பங்களை அடிப்படை. இவை பற்றி நான் கற்றவற்றை கணியம் இதழில் தொடராக எழுதினேன். அவை மின்னூலாகவும் வெளிவருவது மகிழ்ச்சி. எங்கள் திருமண நாளான இன்று இந்த மின்னூல் வெளிவருவது கூடுதல் மகிழ்ச்சி.

தமிழில் கணிணி நுட்பங்களைப் பகிர, ஒரு களமாக உள்ள 'கணியம்' தளத்தில், இதுவரை வெளியான எனது மின்னூல்களுக்கு வாசகர்கள் தரும் ஆதரவு பெருமகிழ்ச்சி அளிக்கிறது.

"தேமதுரத் தமிழோசை உலகெல்லாம் பரவும் வகை செய்தல் வேண்டும்"
"பிற நாட்டு நல்லறிஞர் சாத்திரங்கள் தமிழ் மொழியிற் பெயர்த்தல் வேண்டும்"

என்ற பாரதியின் விருப்பங்களை நிறைவேற்றுவதில், என் பங்களிப்பும் உள்ளது என்பதே, மிகவும் மகிழ்ச்சி.

தொடர்ந்து ஊக்கம் அளிக்கும் என் குடும்பத்தினருக்கும், கணியம் குழுவினருக்கும், FreeTamilEbooks.com குழுவினருக்கும், வாசகர்களுக்கும் நன்றிகள்.



து. நித்யா
கிழக்கு தாம்பரம்
31 அக்டோபர் 2018

மின்னஞ்சல்: nithyadurai87@gmail.com

வலை பதிவு: http://nithyashrinivasan.wordpress.com

[illegible]

இந்த நூலில் உள்ள HTML உதாரணங்கள் யாவும் https://gist.github.com/nithyadurai87 இங்கே உள்ளன.

□□□□□□□□□□□□□

1 JavaScript

JavaScript – □□□□□□□□

JavaScript என்பது ஒரு தனிப்பட்ட நிரலாக்க மொழி கிடையாது. இது html மற்றும் java போன்ற மொழிகளுடன் இணைந்து பயன்படுத்தப்படும் interpreted நிரலாக்க மொழி ஆகும். இது எனவே இதனைக் கற்பதற்கு முன்னர் ' **HTML**' எனும் புத்தகத்தை நன்கு படித்து விடவும். அப்போதுதான் உங்களால் இந்தப் புத்தகத்தில் இடம் பெற்றுள்ள அனைத்தையும் சுலபமாகப் புரிந்து கொள்ள முடியும்.

இது ஒரு Client side scripting language ஆகும். அதாவது வலைத்தளத்தைப் பயன்படுத்தப் போகும் பயன ருடன் தொடர்பு கொள்வதற்கு சிறந்த மொழி. IE, chrome, firefox போன்ற அனைத்து உலாவிகளிலும் இது சிறப்பாகச் செயல்படும். Javascript எப்போதும் HTML program-ன் ஒரு பகுதியாகவே உலாவிகளுக்கு அறிமுகப்படுத்தப்படும். உலாவிகலும் இத்தனை ஒரு HTML program போலவே இயக்கும். எனவே இதற்கென்று தனியாக ஒரு அமைப்பு முறையோ, சேமிப்பு முறையோ கிடையாது.

படிவத்தைப் பூர்த்தி செய்யும் பயனர்கள் இடையில் ஒரு பொத்தானை சொடுக்கும்போதோ அல்லது ஒரு இணைப்பினை சொடுக்கும்போதோ, ஏதோ ஒன்று நிகழும் வகையிலேயே Javascript-ஆனது எப்போதும் எழுதப்படும். அதாவது பயனர்கள் நிகழ்த்தும் விஷயங்களாகவே இது அமையும். எனவே Animation, Multimedia போன்ற இடங்களில் இது பெரிதும் பயன்படும்.

JavaScript -□□ □□□□□□□

<script> tag-ன் பண்புகளான language, type மூலம் நமது javascript-ஆனது html மொழியுடன் இணைக்கப்படுகிறது. இந்த <script>-ஐ html-ல் எங்கு வேண்டுமானாலும் பயன்படுத்தலாம். பொதுவாக <head>-க்குள் காணப்படும். கீழ்க்கண்ட எடுத்துக்காட்டில் <body>-க்குள் பயன்படுத்தியுள்ளோம்.

ஒரு வாக்கியத்தினை வெளிப்படுத்துவதற்கான javascript நிரல் பின்வருமாறு அமையும். இந்த நிரலை gedit அல்லது notepad-ல் எழுதி ஒரு html program-ஐ எவ்வாறு சேமிப்போமோ அவ்வாறே இதனையும் சேமிக்க வேண்டும் .

(Sample.html)

```
<html>
  <body>
    <script language="javascript" type="text/javascript">
      <!--
        document.write("This book will teach you about JavaScript!")
      //-->
    </script>
  </body>
</html>
```

HTML-ன் பழமையான பதிப்புகளில் language=”javascript” type=”text/javascript”

எனும் இரண்டு பண்புகளையும் வெளிப்படையாகக் கொடுக்க வேண்டும். அப்போதுதான் உலாவியானது இதனை javascript-ஆக ஏற்று செயல்படும். ஆனால் தற்போதைய பதிப்புகளில் (XHTML மற்றும் இதையடுத்து வந்தவை) type=”text/javascript” எனும் ஒரு பண்பினை மட்டும் கொடுத்தால் போதுமானது. Language எனும் பண்பினை கொடுக்கத் தேவையில்லை.

அடுத்தபடியாக <script> -க்குள் கொடுக்க வேண்டிய விஷயங்கள் அனைத்தையும் எதற்காக <!– மற்றும் //–> எனும் குறியீடுகளுக்குள் கொடுத்துள்ளோம் என்பதே நீங்கள் முக்கியமாகத் தெரிந்து கொள்ள வேண்டும். இவை இரண்டும் ஒரு பின்னூட்டத்துக்கான (comment) தொடக்கம் மற்றும் முடிவினைக் குறிக்கும் குறியீடுகள் ஆகும். சிலசமயம் javascript-ஐ அடையாளம் கண்டு கொள்ளத் தெரியாத உலாவிகளில் இத்தகைய நிரல்கள் இயங்கும்போது, உலாவியானது குழம்பி ஏதோ தவறு எனும் செய்தியை வெளிப்படுத்தும். இதுவே அவையெல்லாம் ஒரு பின்னூட்டக் குறியீட்டுக்குள் கொடுக்கப்பட்டிருப்பின் javascript-ஐ புரிந்துகொள்ள முடியாவிட்டாலும், இது ஒரு பின்னூட்டம் என்பதைப் புரிந்து கொண்டு அதற்குள் உள்ளவற்றைப் புறக்கணித்து மீதி உள்ளவற்றை செயல்படுத்தும். எனவேதான் javascript நிரலானது எப்போதும் பின்னூட்டக் குறியீட்டுக்குள் கொடுக்கப்படுகிறது.

மேற்கண்ட program-ஐ browser-ல் திறக்கும் பொழுது அது பின்வருமாறு அமைகிறது.

This book will teach you about JavaScript!

2 Variables & Operators in Javascript

Variables

Javascript-ல் உள்ள variable-ஆனது முதல் நிலைத் தரவு வகைகளான (primitive data types) எண்கள், எழுத்துக்கள் மற்றும் ‘true’ , ‘false’ என்பது போன்ற Boolean மதிப்புகளை சேமிக்கும் வல்லமை கொண்டது. மேலும் Null மற்றும் undefined என்பது போன்ற பிற நிலைத் தரவு வகைகளையும் இது ஆதரிக்கும்.

Variable Declaration & Initialization

Javascript-ல் உள்ள ஒரு variable-ஆனது எழுத்துக்கள், எண்கள் மற்றும் தசம எண்கள் போன்ற அனைத்து விதமான தரவுகளையும் தானாகவே அடையாளம் கண்டு கொள்ளும். பிற மொழிகளில் குறிப்பிடுவது போன்று int, flot, string என்றெல்லாம் வெளிப்படையாகச் சொல்லத் தேவையில்லை.

உதாரணம்:

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var salary;
        salary="25000.00";
        var age = 29, name='Nithya';
        document.write(salary,"<br/>",name,"-",age)
      //-->
    </script>
  </body>
</html>
```

ஏதோ ஒரு குறிப்பிட்ட மதிப்பினைத் (25000.00) தாங்கப் போகும் ஒரு பொதுவான வார்த்தைக்கு (salary) variable என்று பெயர். எனவே எந்த வார்த்தையை நாம் variable-ஆகப் பயன்படுத்தப் போகிறோம் என்பதை முன்கூட்டியே அறிவிக்க வேண்டும். இதுவே 'variable declaration' எனப்படும் (var salary). அவ்வாறு அறிவிக்கப்பட்ட variable-க்கு எந்த மதிப்பினை வழங்கப் போகிறோம் என்பதைக் குறிப்பிடும் செயலுக்கு 'variable initialization' என்று பெயர் (salary="25000.00). ஒரு variable-ஐ அறிவிக்கும்போதே அதற்கான மதிப்பினையும் வழங்கலாம் (var age=29). மேலும் ஒன்றுக்கும் மேற்பட்ட variable-களை ஒரே நேரத்தில் நம்மால் அறிவிக்கவும், மதிப்புகளை வழங்கவும் முடியும் (var age = 29, name='Nithya';).

document.write() என்பது அதன் அடைப்புக்குறிக்குள் கொடுக்கப்படும் விஷயங்களை வெளியிட உதவும். இங்கு அடைப்புக்குறிக்குள் (salary,"
",name,"-",age) என்று கொடுக்கப்பட்டிருப்பதால், salary-ன் மதிப்பும்,
 tag-ஆனது cursor-ஐ அடுத்த வரிக்கு அனுப்பி name-ன் மதிப்பையும், Hyphen எனும் குறியீட்டையும், age-ன் மதிப்பையும் வெளிப்படுத்தியுள்ளது.

எனவே இதன் Output பின்வருமாறு அமையும்:

```
25000.00
Nithya-29
```

Local vs Global Variables

அடுத்தபடியாக ஒரே variable-க்கு வெவ்வேறு வகையான மதிப்புகளை அதன் எல்லையைப் பொறுத்து நம்மால் வரையறுக்க முடியும். பின்வரும் உதாரணத்தில் salary எனும் variable-க்கு ஒரு மதிப்பினை (25000.00) அளித்துள்ளோம். பின்னர் அதே variable- ஐ ஒரு function-க்குள் பயன்படுத்தி, வேறு ஒரு மதிப்பினை (50000.00) அளித்துள்ளோம். இவ்வாறு ஒரு சிறிய எல்லையான function-க்குள் வரையறுக்கப்படும் variable-ன் மதிப்பு அதன் எல்லைக்குள்ளேயே முடிந்து விடும். இதனை நாம் "Local variables” (salary=50000.00) என்று கூறுவோம். எந்த ஒரு தனி எல்லைக்குள்ளும் இல்லாமல் பொதுவாக காணப்படுபவை " Global variables” (salary=25000.00) ஆகும்.

```
<html>
  <body onload = checkscope();>
    <script type="text/javascript">
      <!--
        var salary=25000.00;
        document.write(salary);
        function checkscope( )
        {
          var salary = "50000.00";
          document.write(salary);
        }
      //-->
    </script>
  </body>
```

```
</html>
```

இங்கு document.write() மூலம் salary-ன் மதிப்பினை வெளிப்படுத்துமாறு function-க்கு உள்ளேயும், வெளியேயும் கொடுத்துள்ளோம். ஆனால் function-க்கு உள்ளே உள்ள வரி மட்டும் செயல்படுத்தப்பட்டு 50000.00 எனும் மதிப்பினை வெளிப்படுத்தியுள்ளது. எப்போதும் ஒரே variable இருமுறை பயன்படுத்தப்பட்டிருப்பின், local-ஆக அமைக்கப்பட்ட variable-ன் மதிப்பே முதன்மை பெறும்.

```
50000.00
```

எனவே இதன் Output பின்வருமாறு அமையும்:

Reserved Words

ஒரு சில வார்த்தைகளை (eg: export, final, long) நம்மால் variable-ஆக அறிவிக்க முடியாது. ஏனென்றால் இது போன்ற வார்த்தைகளுக்கு என்னென்ன மதிப்புகள் என்பதை javascript-ஐ உருவாக்கியவர்களே நியமித்து விட்டனர். அவையே ‘Reserved Keywords’ எனப்படும். இதன் பட்டியல் பின்வருமாறு:

abstract
boolean
break
byte
case
catch
char
class
const
continue
debugger
default
delete
do
double

else
enum
export
extends
false
final
finally
float
for
function
goto
if
implements
import
in

instanceof
int
interface
long
native
new
null
package
private
protected
public
return
short
static
super

switch
synchronized
this
throw
throws
transient
true
try
typeof
var
void
volatile
while
with

3 Operators

Operator என்றால் இயக்குபவர் என்று பொருள். கொடுக்கப்பட்ட மதிப்புகளைப் பெற்றுக்கொண்டு, அதன் மீது ஒருசில செயல்பாடுகளை செலுத்தி, நமக்கு வேண்டிய தகவல்களை வெளிப்படுத்தும் வேலையை செய்பவருக்கு operator என்று பெயர். Javascript-ல் arithmetic, comparison, logical, conditional, typeof, assignment போன்ற பல்வேறு வகையான operators உள்ளன.

கீழ்க்காணும் program-ல் அனைத்து வகை operator- களின் செயல்பாடுகளும் ஒரு சேரக் கொடுக்கப்பட்டுள்ளது.

```
<html>
  <body>
    <script type="text/javascript">
      <!--
        var a = 100;
        var b = 40;

        document.write("Arithmetic - ");
        document.write(a%b);
        document.write("<br />");
```

```
        document.write("Comparison - ");
        document.write(a<b);
        document.write("<br />");

        document.write("Logical - ");
        document.write(((b<=100)&&(a>=100)));
        document.write("<br />");

        document.write("Conditional - ");
        document.write((a > b) ? a : b);
        document.write("<br />");

        document.write("typeof - ");
        document.write(typeof a=="string" ? "The value is string" : "The value
is not a string");
        document.write("<br />");

        document.write("Assignment - ");
        document.write((a -= b));
        document.write("<br />");
      //-->
    </script>
  </body>
</html>
```

இதன் Output பின்வருமாறு:

```
Arithmetic - 20
Comparison - false
Logical - true
Conditional - 100
typeof - The value is not a string
Assignment - 60
```

இப்போது மேற்கண்ட program- க்கான விளக்கத்தையும், ஒவ்வொரு வகை operator-ன் கீழும் வரும் அனைத்து விதமான குறியீடுகளின் செயல்பாடுகளையும் பின்வருமாறு காணலாம். மேற்கண்ட அதே program-ல் document.write() எனும் வரியை மட்டும் கீழே கொடுக்கப்பட்டுள்ளவாறு மாற்றி மாற்றி இயக்கி பார்க்கவும்.

Arithmetic Operators

இதன் கீழ் வரும் கூட்டல், கழிதல், பெருக்கல், வகுத்தல் என்னென்ன செய்யும் என்பது உங்களுக்கே தெரியும். இதனுடன் சேர்த்து Modulus (%), Increment(++), Decrement (–) எனும் 3 வகை குறியீடுகள் உள்ளன. % என்பது இரண்டு எண்களை வகுத்தல் கிடைக்கும் மீதியை வெளிப்படுத்தும். ++ என்பது கொடுக்கப்பட்ட எண்ணுடன் ஒரு எண்ணைக் கூட்டியும், — என்பது கொடுக்கப்பட்ட எண்ணுடன் ஒரு எண்ணைக் கழித்தும் வெளிப்படுத்தும்.

Arithmetic Operators	
document.write(a+b)	140
document.write(a-b)	60
document.write(a*b)	4000
document.write(a/b)	2.5
document.write(a%b)	20
document.write(++a)	101
document.write(--b)	39

Comparison Operators

இரண்டு எண்களை ஒப்பிட்டு அவற்றுக்குள் எது பெரியது (>), சிறியது (<), சமமானது (==), சமமில்லாதது (!=), என்பது போன்ற தகவல்களை வெளிப்படுத்த உதவும். (>=) என்பது சமமானது அல்லது பெரியது என்றும், (<=) என்பது சமமானது அல்லது சிறியது என்றும் பொருள்படும். இதன் வெளிப்பாடு எப்போதும் உண்மை (true), பொய் (false) எனும் இரண்டு மதிப்புகளையே பெற்றிருக்கும்.

Comparison Operators	
document.write(a == b)	FALSE
document.write(a!=b)	TRUE
document.write(a>b)	TRUE
document.write(a<b)	FALSE
document.write(a>=b)	TRUE
document.write(a<=b)	FALSE

-

Logical (or Relational) Operators

இது உண்மை, பொய் எனும் 2 மதிப்புகளை ஒப்பீடு செய்யப் பயன்படுகிறது. && ஒப்பிடப்படுகின்ற அனைத்தும் உண்மை என்பதை வெளிப்படுத்தினால் மட்டுமே, இதுவும் உண்மை என்பதை வெளிப்படுத்தும். !! ஒப்பிடப்படுபவைகளில் ஏதேனும் ஒன்றின் வெளிப்பாடு உண்மை என்று இருந்தால் கூட இதுவும் உண்மை என்பதை வெளிப்படுத்தி விடும். ! என்பது வெளிப்படும் விடை உண்மையாக இருப்பின் பொய் என்றும், பொய்யாக இருப்பின் உண்மை என்றும் மாற்றி வெளிப்படுத்தும்.

-

Logical Operators	
document.write(((b<=100)&&(a>=100)))	TRUE
document.write(((b<=100)||(a<=100)))	TRUE
document.write(!((b<=100)&&(a>=100)))	FALSE

-

Assignment Operators

= என்பது வலப்புறம் இருக்கும் மதிப்பினை, இடப்புறம் உள்ள variable-க்கு வழங்கிவிடும். b=40 என்று ஒரு variable காணப்படின், a=b எனக் கொடுக்கும்போது, a-ன் மதிப்பு 40 என மாறிவிடும். பின்னர் a+=b எனக் கொடுக்கும்போது, ஏற்கனவே a-ல் உள்ள 40 எனும் மதிப்பு, b-ல் உள்ள 40-வுடன் கூட்டப்பட்டு a-ன் மதிப்பு 80 என மாறிவிடும். a-=b எனக் கொடுக்கும்போது 80-லிருந்து 40 கழிக்கப்பட்டு மீண்டும் a-ன் மதிப்பு 40-க்கே வந்துவிடும். இப்போது பின்வரும் உதாரணத்தைப் பார்த்தால் உங்களாலேயே புரிந்து கொள்ள முடியும்.

Assignment Operators	
document.write(a = b)	40
document.write(a+=b)	80
document.write(a-=b)	40
document.write(a*=b)	1600
document.write(a/=b)	40
document.write(a%=b)	0

Miscellaneous Operator

Conditional (or ternary) Operators (a > b) ? a : b என்பது ஒரு சோதனையை செய்து, அதன் வெளீப்பாடு உண்மை என இருப்பின், ?-க்கு அடுத்து உள்ள மதிப்பினையும், பொய் என இருப்பின் அந்த மதிப்பிற்கு அடுத்து உள்ள மதிப்பினையும் வெளிப்படுத்தும்.

(typeof a==”string” ? "xxx”: "yyy”); என்பது ஒரு variable-ல் உள்ள மதிப்பின் வகையை சோதனையை செய்து, அதன் வெளீப்பாடு உண்மை என இருப்பின், ?-க்கு அடுத்து உள்ள மதிப்பினையும், பொய் என இருப்பின் அந்த மதிப்பிற்கு அடுத்து உள்ள மதிப்பினையும் வெளிப்படுத்தும்.. அதாவது ஒரு variable-ஆனது "number”, "string”, "Boolean”, "object”, "function”, "undefined” போன்ற data வகைகளுடன் ஒப்பிடப்படுகிறது.

Miscellaneous Operators	
document.write((a > b) ? a : b)	100
document.write(typeof a=="string" ? "The value is string”: "The value is not a string");	The value is not a string

4 Conditional and Looping Statements in javascript

Conditional statements

ஒரு variable-ல் சேமிக்கப்பட்டுள்ள மதிப்பானது பல்வேறு நிபந்தனைகளோடு ஒப்பிடப்படும். ஒவ்வொரு நிபந்தனையும் பல்வேறு வகையான நிகழ்வுகளைக் கொண்டிருக்கும். ஒப்பிடப்படுகின்ற மதிப்பானது எந்த நிபந்தனையோடு ஒத்துப்போகிறதோ, அதனுடைய நிகழ்வினை நிகழ்த்தும் செயலுக்கு If…Else மற்றும் switch_case போன்ற conditional statements பயன்படுகின்றன.

If…Else

பின்வரும் உதாரணத்தில் age எனும் variable-ல் உள்ள மதிப்பு 18 முதல் 21 வரை இருப்பின் "Legally fit for marriage" எனும் வாக்கியத்தையும், 21-ஐ விட அதிகமாக இருந்தால் "Legally and Medically fit" எனும் வாக்கியத்தையும், வேறு எந்த மதிப்பாக இருந்தாலும் "Not fit for marriage" எனும் வாக்கியத்தையும் வெளிப்படுத்துமாறு கொடுக்கப்பட்டுள்ளது.

ஒரு If…Else சோதனையில் எப்போதும் if, else if, else எனும் 3 வகையான சோதனைகளை கட்டாயம் கொடுக்க வேண்டும் என்று அவசியமில்லை. If மட்டுமே கொடுத்து ஒரு சோதனையுடன் நிறுத்திக்கொள்ளலாம். If-வுடன் சேர்த்து Else-ஐயும் கொடுத்து இரண்டு சோதனையுடன் நிறுத்திக்கொள்ளலாம். பல்வேறு சோதனைகளை நிகழ்த்த விரும்பினால் else if -மூலம் வேண்டிய எண்ணிக்கையில் சோதனைகளை சேர்த்துக்கொண்டேயும் செல்லலாம்.

```
<html>
  <body>
    <script type="text/javascript">
      <!--
        var age = 24;
        if((age >= 18) && (age <= 21)){document.write("Legally fit for
marriage");}
```

```
        else if( age > 21 ){document.write("Legally and Medically fit for marriage");}
        else{document.write("Not fit for marriage");}
      //-->
    </script>
  </body>
</html>
```

இதன் Output பின்வருமாறு:

Legally and Medically fit

Switch Case

பின்வரும் உதாரணத்தில் switch- க்குள் ஒரு குறிப்பிட்ட மதிப்பினைத் (II) தாங்கியுள்ள variable (grade) கொடுக்கப்பட்டுள்ளது. அதன்கீழ் உள்ள case-க்குள் பல்வேறு மதிப்புகளும் அதற்கான வெளிப்பாடுகளும் கொடுக்கப்பட்டுள்ளன. Switch-க்குள் உள்ள variable-ன் மதிப்பானது case-க்குள் உள்ள மதிப்புகளுடன் ஒப்பிடப்பட்டு, எந்த மதிப்புடன் ஒத்துப்போகிறதோ (case ‘II’), அதற்கான வெளிப்பாட்டினை ("Pretty good”) நிகழ்த்துகிறது. இந்த வகையான செயல்பாட்டிற்கு நாம் பல்வேறு Else If கூட பயன்படுத்தலாம். இவை இரண்டிற்கும் உள்ள ஒரே ஒரு வித்தியாசம் என்னவெனில் Swtich Case-ஆனது ஒரே ஒரு variable-ன் மதிப்பினை மட்டுமே பல்வேறு மதிப்புகளுடன் ஒப்பிடும்; Else If- ஆனது ஒன்றுக்கும் மேற்பட்ட variable-களை இணைத்து பல்வேறு மதிப்புகளுடன் ஒப்பிடும்.

```
<html>
```

```
  <body>
    <script type="text/javascript">
      <!--
        var grade='II';
        switch (grade)
        {
          case 'I': document.write("Good job");break;
          case 'II': document.write("Pretty good");break;
          case 'III': document.write("Passed");break;
          case 'IV': document.write("Not so good");break;
          case 'V': document.write("Failed");break;
          default:  document.write("Grade is not valid")
        }
      //-->
    </script>
  </body>
</html>
```

இதன் Output பின்வருமாறு:

Pretty good

5 Looping statements

ஒரு குறிப்பிட்ட நிரல் தொகுப்பினை மீண்டும் மீண்டும் இயக்குவதற்கு while, for போன்றவை பயன்படுகின்றன. இவை இரண்டும் முதலில் ஒரு variable-ன் அடிப்படையில் condition-ஐ வலியுறுத்தும். பின்னர் அந்த condition பொய்யாகும் வரை variable- ன் மதிப்பினை ஒவ்வொன்றாக அதிகரிப்பதன் மூலம் சுழற்சியை உருவாக்குகின்றன.

While & Do while loop

இதில் while மற்றும் do while எனும் இரண்டு வகைகள் உள்ளன. கீழ்க்கண்ட எடுத்துக்காட்டில் x-ஆனது while-ன் செயல்பாட்டினை விளக்குவதற்கும், y-ஆனது do while-ன் செயல்பாட்டினை விளக்குவதற்கும் பயன்பட்டுள்ளது. இவை இரண்டுக்கும் முதலில் 0 எனும் மதிப்பு வழங்கப்பட்டுள்ளது.

While loop: இது X- ன் மதிப்பு 10-க்கும் குறைவாக இருக்கும்வரை அடைப்புக் குறிக்குள் { } உள்ளவற்றை செயல்படுத்துகிறது. முதலில் Number:0 என்பது வெளிப்படுகிறது. பின்னர் X++ என்பது x-ன் மதிப்புடன் 1-ஐ கூட்டி அடுத்த சுழற்சியைத் தூண்டுகிறது. எனவே அடுத்த சுழற்சியில், Number:1 என்பதும் வெளிப்படும். எப்போது x-ன் மதிப்பு 10-ஐ அடைகிறதோ அப்போது அடைப்புக் குறிக்குள் { } உள்ளவற்றை செயல்படுத்தாமல் சுழற்சியை நிறுத்துகிறது.

Do while loop: do-வைத் தொடர்ந்து என்ன செயல்படுத்த வேண்டுமோ, அதை முதலில் கொடுத்து விட வேண்டும். அதன் பின்னர் condition வலியுறுத்தப்படும். இங்கு அடைப்புக் குறிக்குள் { } உள்ளவை y- ன் மதிப்பு 5-க்கும் குறைவாக இருக்கும் வரை செயல்படுத்தப்படுகின்றன. இவை இரண்டும் பார்ப்பதற்கு ஒரே

மாதிரியாக இருந்தாலும், அதன் செயல்முறையில் வேறுபடுகின்றன.

While ஆனது முதலில் சோதனை பொருந்துகிறதா எனப் பார்த்து, பின்னர் அடைப்புக்குறிக்குள் உள்ளவற்றை செயல்படுத்துவதால் இது ‘’Entry control loop’ எனப்படும்.

Do While ஆனது அடைப்புக்குறிக்குள் உள்ளவற்றை செயல்படுத்திய பின்னரே சோதனை பொருந்துகிறதா எனப் பார்ப்பதால், இது ‘Exit control loop’ எனப்படும். எனவே Do while-ல் சோதனை பொருந்தவில்லை என்றாலும், அதனால் அடுத்த சுழற்சியை மட்டுமே தடுக்க முடியுமே தவிர, ஏற்கனவே அடைப்புக்குறிக்குள் நடந்து முடிந்த ஒன்றை தவிர்க்க இயலாது.

```
<html>
  <body>
    <script type="text/javascript">
      <!--
        var x = 0;
        var y = 0;

        while (x < 10){document.write("Number:" + x + "<br />");x++;}

        do{document.write(y);y++;} while (y < 5);
      //-->
    </script>
  </body>
</html>
```

```
Number:0
Number:1
Number:2
Number:3
Number:4
Number:5
Number:6
Number:7
Number:8
Number:9
01234
```

இதன் Output பின்வருமாறு:

For loop

For loop- ம் while போன்றதே. இதன் வடிவம் மட்டுமே மாறுபட்டுள்ளது. ஒரே ஒரு வித்தியாசம் என்னவெனில், loop-க்குள் variable-ன் துவக்க மதிப்பு வரையறுக்கப்படுகிறது. கீழ்க்கண்ட எடுத்துக்காட்டில் for loop- ஆனது x-ன் மதிப்பு 0 என்றும், அது 10-க்கும் குறைவாக (x<10) இருக்கும் வரை அடைப்புக் குறிக்குள் { } உள்ளவற்றை செயல்படுத்த வேண்டும் எனவும், ஒவ்வொரு முறையும் துவக்க மதிப்புடன் 1 கூட்டப்பட வேண்டும் (x++) எனவும் கூறுகிறது.

For…In என்பது for loop-ன் மற்றொரு வகை. இது variable-ஐ ஒரு object-ன் வழியே செலுத்தி அதன் அனைத்து பண்புகளையும் ஒவ்வொன்றாக வெளிக்காட்ட உதவுகிறது. X எனும் variable- ஐ கீழ்க்கண்டவாறு navigator எனும் object வழியே செலுத்தும்போது அது அதன் அனைத்து பண்புகளையும் ஒவ்வொன்றாக வெளிப்படுத்துவதைக்
காணலாம்.

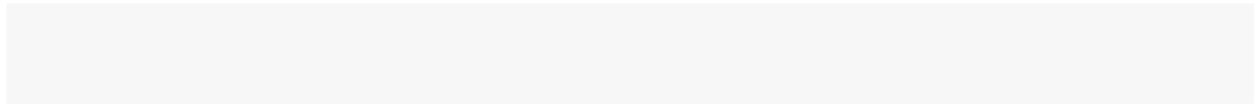
```
<html>
  <body>
    <script type="text/javascript">
      <!--
        var x;

        for(x = 0; x < 10; x++){document.write(x);}

        document.write("<br />");

        for (x in navigator) {document.write(x+"<br />");}
      //-->
```

```
    </script>
  </body>
</html>
```



இதன் Output பின்வருமாறு:

```
0123456789
vendorSub
productSub
vendor
maxTouchPoints
hardwareConcurrency
appCodeName
appName
appVersion
platform
product
userAgent
language
languages
onLine
cookieEnabled
doNotTrack
geolocation
mediaDevices
plugins
mimeTypes
webkitTemporaryStorage
webkitPersistentStorage
serviceWorker
getBattery
sendBeacon
requestMediaKeySystemAccess
getGamepads
```

Break…Continue

ஒரு சுழற்சி நடந்துகொண்டிருக்கும்போது, இடையில் ஒரு குறிப்பிட்ட சுழற்சி எண்ணின் போது அச்சுழற்சியிலிருந்து மொத்தமாக வெளியேறுவதற்கு break-ம், அச்சுழற்சி எண்ணின் போது மட்டும் அடைப்புக்குறிக்குள் உள்ளவற்றைத் தவிர்த்து, அடுத்த சுழற்சி எண்ணிலிருந்து ஆரம்பிப்பதற்கு continue-ம் பயன்படுகின்றன. Continue- வை பயன்படுத்துவதற்கு சுலபமாக, ஒன்றுக்கும் மேற்பட்ட சுழற்சிகள் காணப்படின், ஒரு குறிப்பிட்ட சுழற்சியின் அருகில் அமைக்கப்படும் identifier-தான் Labels ஆகும்.

கீழ்க்கண்ட எடுத்துக்காட்டில் x-ஆனது break-ன் செயல்பாட்டினை விளக்குவதற்கும், y-ஆனது continue-ன் செயல்பாட்டினை விளக்குவதற்கும் பயன்பட்டுள்ளது. Heading: என்பது ஆகும். x, y இரண்டுக்கும் முதலில் 0 எனும் மதிப்பு வழங்கப்பட்டுள்ளது.

Break: x- ன் மதிப்பு 10-க்கும் குறைவாக இருக்கும் வரை அடைப்புக் குறிக்குள் { }
உள்ளவற்றை செயல்படுத்துமாறு ஒரு while loop அமைக்கப்பட்டுள்ளது. இதில் x-ன் மதிப்பானது 5 எனும் சுழற்சி எண்ணை அடையும்போது சுழற்சியை விட்டு வெளியேதும் வகையில் if (x == 5){break;} எனும் வாக்கியம் கொடுக்கப்பட்டுள்ளது.

Continue: y- ன் மதிப்பு 10-க்கும் குறைவாக இருக்கும் வரை அடைப்புக் குறிக்குள் { }
உள்ளவற்றை செயல்படுத்துமாறு ஒரு while loop அமைக்கப்பட்டுள்ளது. இதில் y-ன் மதிப்பானது 5 எனும் சுழற்சி எண்ணை அடையும்போது மட்டும் if (y == 5){continue;}, அடைப்புக்குறிக்குள் உள்ளவற்றை செயல்படுத்தாமல் loop-ன் தொடக்கத்தை சென்றடைந்து அடுத்த சுழற்சி எண்ணிலிருந்து சுழல ஆரம்பிக்கிறது.

Labels: i-ன் மதிப்பு 3-க்கும் குறைவாக இருக்கும் வரை வெளிப்புற loop சுழலும்படியும், அதற்குள் உள்ள j-ன் மதிப்பு 5-க்கும் குறைவாக இருக்கும் வரை உட்புற loop சுழலும்படியும் 2 for loops அமைக்கப்பட்டுள்ளன. உட்புற loop-ல் j-ன் மதிப்பு 3-ஐ அடையும்போது Heading எனும் பெயர் கொண்ட label உள்ள இடத்திற்குச் செல்லுமாறு ஒரு வரி if (j == 3){continue Heading;} கொடுக்கப்பட்டுள்ளது. இந்த Heading: வெளிப்புற loop-ன் துவக்கத்தில் கொடுக்கப்பட்டிருப்பதால், அங்கிருந்து சுழற்சி எண் 3- க்கான சுழற்சியை ஆரம்பிக்கும். ஆனால் (i<3) எனும் சோதனைக்குள் அடங்காததால், இங்கு சுழற்சி முடிக்கப்பட்டுள்ளது.

```
<html>
  <body>
    <script type="text/javascript">
      <!--
      var x = 1;
      var y = 1;
      while (x < 10)
      {
        if (x == 5){break;}
        x = x + 1;
        document.write( x + "<br />");
      }

      document.write("<br />");

      while (y < 10)
      {
        y = y + 1;

        if (y == 5){continue;}
```

```
          document.write(y);
        }

        document.write("<br />");

        Heading:

        for (var i = 0; i < 3; i++)
        {
          document.write(i + "times" + "<br />");
          for (var j = 0; j < 5; j++)
          {
            if (j == 3){continue Heading;}
            document.write("Value is: " + j + "<br />");
          }
        }
        //-->
      </script>
   </body>
</html>
```

இதன் Output பின்வருமாறு:

2
3
4
5

234678910
0times
Value is: 0
Value is: 1
Value is: 2
1times
Value is: 0
Value is: 1
Value is: 2
2times
Value is: 0
Value is: 1
Value is: 2

6 Functions & Events in JavaScript

Functions & Events

Functions என்பது மறுபயன்பாட்டிற்கு உதவும் வகையில் எழுதப்படுகின்ற நிரல்கள் ஆகும். ஒரு மிகப்பெரிய program- ஐ நாம் எழுதிக் கொண்டிருக்கும்போது ஒருசில குறிப்பிட்ட நிரல்களை மட்டும் நமது தேவைக்கேற்ப நாம் மீண்டும் மீண்டும் பயன்படுத்த வேண்டியிருக்கும். அப்படிப்பட்ட நிரல்களை ஒரு பொதுவான பெயர் வைத்து சேமித்துக்கொள்ள functions பயன்படுகிறது. சுருக்கமாகச் சொல்லப்போனால், ஒரு மிகப்பெரிய program-ஐ சிறுசிறு பகுதிகளாகப் பிரித்துக்கையாளுவதற்கு functions பயன்படுகிறது.

இதை parameters ஏற்றுக்கொண்டு செயல்படுபவை, parameters இல்லாமல் செயல்படுபவை என்று இரண்டாகப் பிரிக்கலாம். பின்வரும் எடுத்துக்காட்டில் sayHello() என்பது parameters இல்லாமல் செயல்படும் function-க்கு உதாரணமாகவும், concatenate() என்பது parameters வைத்து செயல்படும் function-க்கு உதாரணமாகவும் விளங்குகிறது.

```
<html>
  <head>
    <script type="text/javascript">
      function sayHello()
      {
        document.write ("Hello Buddy! Welcome!");
      }

      function concatenate(fn, ln)
      {
        var name;
        name = fn + ln;
```

```
            return name;
         }

         function fullname()
         {
            var c;
            c = concatenate('Malathi', 'Sivakumar');
            document.write (c );
         }
      </script>
   </head>

   <body>
      <form>
         <input type="button" onclick="sayHello()" value="Greetings">
         <input type="button" onclick="fullname()" value="Click Here">
      </form>
   </body>

</html>
```

இதன் Output பின்வருமாறு:



Hello Buddy! Welcome!   MalathiSivakumar

Functions without parameters

sayHello() எனும் பெயரில் சேமிக்கப்பட்ட function, "Hello Buddy! Welcome!" எனும் வரியினை வெளிப்படுத்தும் வேலையை செய்கிறது. பின்னர் ‘Greetings’ எனும் பெயர் கொண்ட பொத்தானை சொடுக்கும்போது, இந்த function அழைக்கப்படும் வகையில் body-க்குள் ஒரு HTML code கொடுக்கப்பட்டுள்ளது. இது parameter இல்லாமல் அழைக்கப்படும் function-க்கு உதாரணம் ஆகும்.

Functions with parameters

"Click Here" எனும் பொத்தானை சொடுக்கும்போது, fullname() எனும் function அழைக்கப்படுவதற்கான code ஒன்று body-க்குள் உள்ளது. fullname()-க்குள் c எனும் ஒரு variable-வரையறுக்கப்பட்டு அதில், concatenate() function வெளிப்படுத்தும் மதிப்பு சேமிக்கப்படுகிறது. இந்த concatenate()தான் parameters-ஐ ஏற்றுக்கொண்டு செயல்படும் function-க்கு உதாரணம் ஆகும்.

அதாவது c = concatenate('Malathi', 'Sivakumar') எனக் கொடுப்பதன் மூலம், அடைப்புக்குறிக்குள் உள்ள மதிப்புகள் concatenate(fn, ln) என்று வரையறுக்கப்பட்டுள்ள function-க்குள் உள்ளீடு மதிப்புகளாகச் சென்று அவற்றுக்குள் உள்ளவற்றைச் செயல்படுத்தும்.

concatenate(fn, ln)-க்குள் name எனும் ஒரு variable-வரையறுக்கப்பட்டு அதில், fn, ln வழியே செலுத்தப்பட்ட 2 தனித்தனி மதிப்புகள் இணைக்கப்பட்டு MalathiSivakumar எனும் ஒரே மதிப்பாக சேமிக்கப்படும். பின்னர் இந்த மதிப்பு சேமிக்கப்பட்டுள்ள name எனும் variable, return keyword மூலம் வெளியே அனுப்பப்படுகிறது. இவ்வாறு வெளியே வந்துள்ள இந்த மதிப்பே c எனும் variable-ல் சேமிக்கப்பட்டு வெளிப்படுத்தப்படுகிறது.

Events

ஒரு விஷயம் browser-ல் நடப்பதற்கு நாம் செய்யும் ஒவ்வொன்றும் Events எனப்படும். ஒரு பொத்தானை சொடுக்குவது, Mouse-ஐ நகர்த்துவது, Page-ஐ load செய்வது போன்ற ஒவ்வொன்றும் ஒவ்வொரு Events ஆகும்.

மேற்கண்ட அதே Program- ஐ நாம் Events-க்கும் எடுத்துக்கொள்ளலாம். அதில் onclick என்பது ஒரு event. அதையே onmouseover, onmouseout எனும் events கொண்டு மாற்றி இயக்கிப் பார்க்கவும்.

```
<html>

  <head>
    <script type="text/javascript">
      function sayHello()
      {
        document.write ("Hello Buddy! Welcome!");
      }

      function concatenate(fn, ln)
      {
        var name;
        name = fn + ln;
        return name;
      }

      function fullname()
      {
        var c;
```

```
        c = concatenate('Malathi', 'Sivakumar');
        document.write (c );
      }
    </script>
  </head>

  <body>
    <form>
      <input type="button" onmouseover="sayHello()" value="Greetings">
      <input type="button" onmouseout="fullname()" value="Click Here">
    </form>
  </body>

</html>
```

onmouseover=”sayHello()”எனும்போது "Greetings” பொத்தானின் அருகில் mouse-ஐ கொண்டு செல்லும் போதே sayHello() இயக்கப்பட்டுவிடும்.

onmouseout=”fullname()”எனும்போது "Click Here” பொத்தானின் அருகில் mouse-ஐ கொண்டு செல்லும் போது எதுவும் நிகழாது. ஆனால் அங்கிருந்து mouse-ஐ வெளியே நகர்த்தும்போது sayHello() இயக்கப்பட்டுவிடும்.

7 Window-□□□□ □□□□□□□□□

window எனும் object-ன் ஒருசில பண்புகளைப் பயன்படுத்தி நாம் செய்யும் வேலைகளைப் பின்வருமாறு காணலாம்.

location

ஒரு வலைத்தளத்தை நாம் பயன்படுத்திக் கொண்டிருக்கும்போது, திடீரென்று ‘Home’ எனும் பொத்தானை அழுத்தி, அதன் முதல் பக்கத்திற்குச் செல்வோம் அல்லவா, இதற்காக window.location எனும் பண்பு பயன்படுகிறது. அதாவது browser-ன் எந்தப் பக்கத்தில் பயனர் இருந்தாலும், ஒரு பொத்தானை அழுத்தினால், ஒரு குறிப்பிட்ட url-க்குச் செல்லுமாறு அமைக்க இந்தப் பண்பு பயன்படுகிறது.

கீழ்க்கண்ட எடுத்துக்கத்தில், window.location-ன் மதிப்பு www.kaniyam.com என்று கொடுக்கப்பட்டுள்ளது. பின்னர் ‘Redirect Me’ எனும் பொத்தானை சொடுக்கும்போது, இது செயல்படும் வகையில் ஒரு HTML code கொடுக்கப்பட்டுள்ளது.எனவே வலைத்தளப் பக்கத்தில் ‘Redirect Me’ எனும் பொத்தானை எங்கு வைத்தாலும், அந்தப் பொத்தானை அழுத்தும் போது, அது www.kaniyam.com எனும் முகவரிக்குச் சென்று விடும். இது "Page Redirection” எனப்படும்.

```
<html>

  <head>

    <script type="text/javascript">
```

```
        <!--
          function Redirect()
          {
             window.location="http://www.kaniyam.com";
          }
        //-->
     </script>

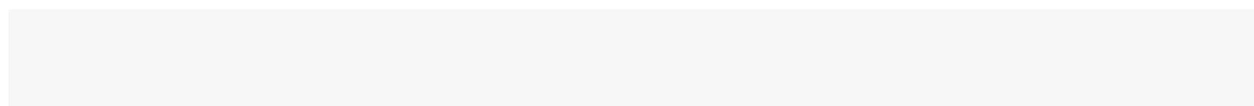
  </head>

  <body>

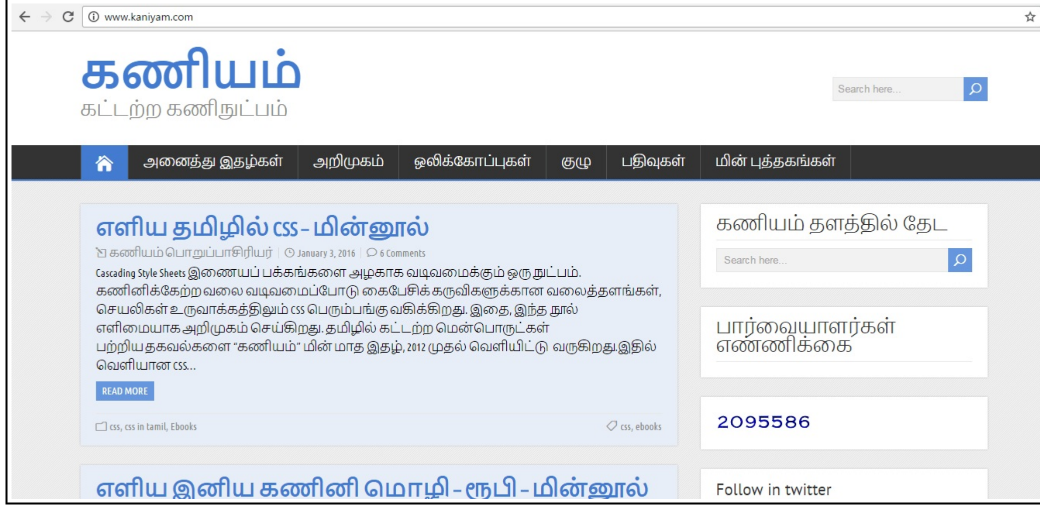
     <form>
        <input type="button" onclick="Redirect()" value="Redirect Me">
     </form>

  </body>

</html>
```







இதன் Output பின்வருமாறு:

Print

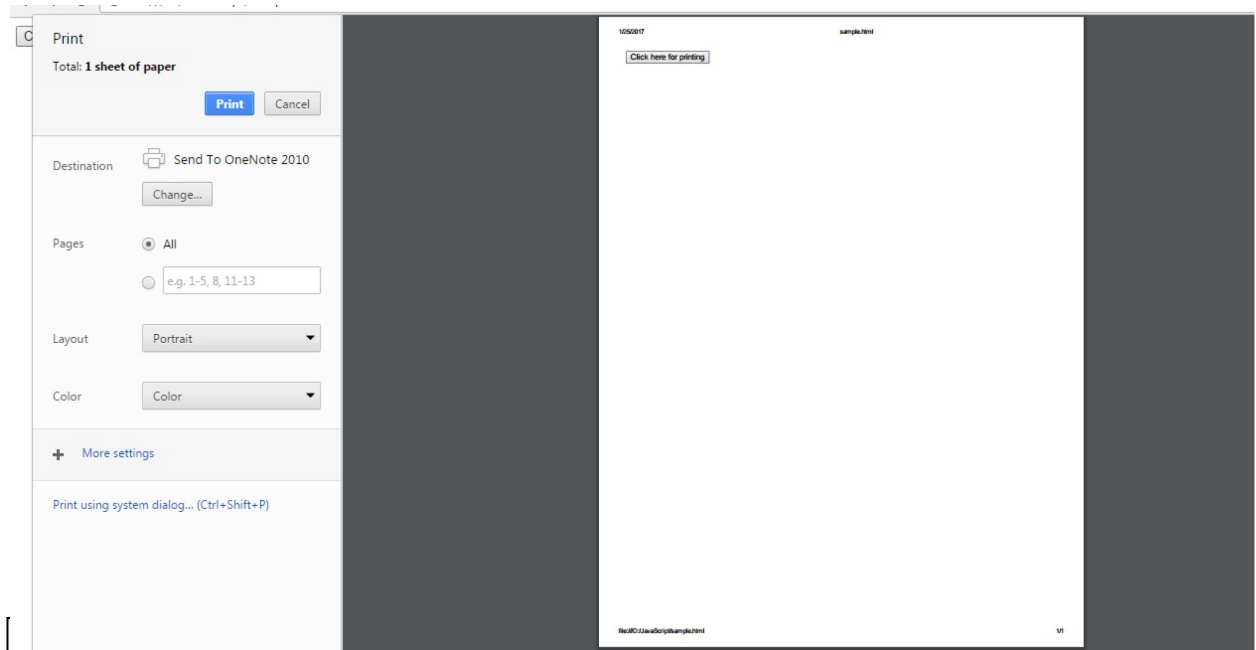
window.print() என்பது தற்போதைய வலைப் பக்கத்தை அச்சிட உதவும் ஒரு print window-வை உருவாக்கும். கீழ்க்கண்ட program-ல் "Click here for printing" எனும் பெயர் கொண்ட ஒரு பொத்தானை சொடுக்கும்போது இத்தகைய window-வை உருவாக்கும் வகையில் ஒரு html code கொடுக்கப்பட்டுள்ளது.

```
<html>
  <body>
    <form>
      <input type="button" onclick="window.print()" value="Click here for
```

printing">
</form>
</body>
</html>

Click here for printing

இதன் Output பின்வருமாறு:



8 Dialog Boxes and Exception Handling

Dialog Boxes

Javascript-ல் 3 முக்கியமான பெட்டிகள் உள்ளன. அவற்றைக் கீழ்க்காணும் எடுத்துக்காட்டில் காணலாம்.

- "Alert box” எனும் பெயர் கொண்ட பொத்தானின் மீது சொடுக்கும்போது "This is a warning message!” எனும் செய்தி வெளிப்படும் வகையில் ஒரு பெட்டி உருவாக்கப்பட்டுள்ளது. இது பயனர்களை எச்சரிக்க உதவும் alert() பெட்டி ஆகும்.

- "Confirm box” எனும் பெயர் கொண்ட பொத்தானின் மீது சொடுக்கும்போது "Do you want to continue?” என்ற ஒரு கேள்வியைக் கேட்டு, அதற்கு ஆம்/இல்லை என்று நாம் பதிலளிக்கும் வகையில் ஒரு பெட்டி உருவாக்கப்பட்டுள்ளது. இது confirm() பெட்டி எனப்படும்.

- "prompt box” எனும் பெயர் கொண்ட பொத்தானின் மீது சொடுக்கும்போது "Please enter your name:” எனக் கூறி அதற்கு நாம் அளிக்கும் விடையைப் பெற்றுக்கொள்ளும் வகையில் ஒரு பெட்டி உருவாக்கப்பட்டுள்ளது. இது பயனர்கள் கொடுக்கும் விவரங்களை உள்ளீடாகப் பெற்றுக்கொண்டு, அதற்கேற்ப செயல்படும் prompt() பெட்டி ஆகும்.

```
<html>
  <head>
    <script type="text/javascript">
      <!--
        function box1()
        {
          alert ("This is a warning message!");
```

```
          }
          function box2()
          {
            var x = confirm("Do you want to continue ?");
            if( x == true ){document.write ("User wants to continue!");}
            else{document.write ("User does not want to continue!");}
          }
          function box3()
          {
            var x = prompt("Please enter your name : ", "your name goes here");
            document.write("Hello! Welcome " + x);
          }
        //-->
      </script>
    </head>

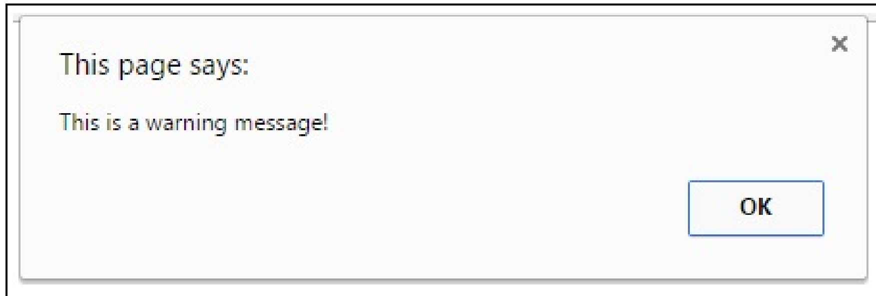
    <body>
      <form>
        <input type="button" onclick="box1()" value="Alert box">
        <input type="button" onclick="box2()" value="Confirm box">
        <input type="button" onclick="box3()" value="prompt box">
      </form>
    </body>
</html>
```

இதன் வெளிப்பாடு பின்வருமாறு:

Alert box | Confirm box | prompt box

alert():

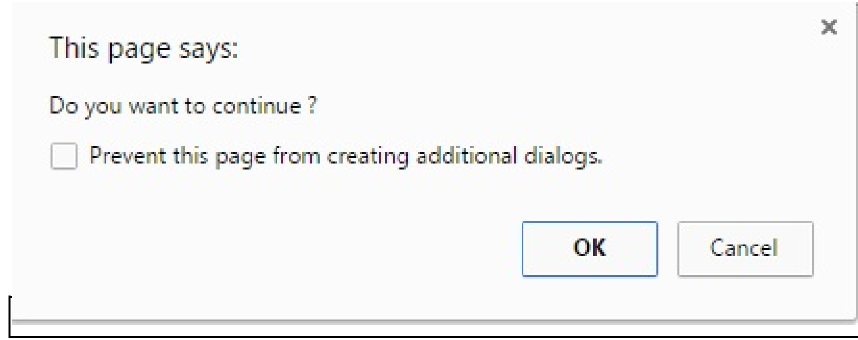
பயனர்களை எச்சரிக்கும் வகையில் ஒரு பெட்டியை திரையில் வெளிப்படுத்த alert() box பயன்படும். இந்த alert()-க்குள் கொடுக்கப்படும் செய்தியானது ("This is a warning message!") திரையில் வெளிப்படும் பெட்டிக்குள் எச்சரிப்பு செய்தியாக



இருக்கும். பெட்டிக்குள் உள்ள OK பொத்தானின் மீது சொடுக்கினால் அந்த எச்சரிப்பு செய்தி மறைந்துவிடும்.

confirm():

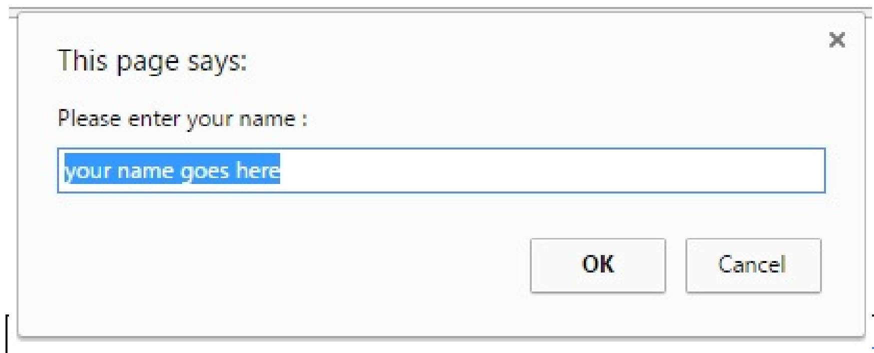
பயனர்களிடமிருந்து ஒரு முறைக்கு இருமுறை உறுதியைப் பெற்றுக்கொள்ளும் வகையில் ஒரு பெட்டியை திரையில் வெளிப்படுத்த confirm() box பயன்படும். ஏனெனில் முதல்முறை தவறுதலாகச் சொடுக்கியிருந்தாலும் கூட இந்தப் பெட்டியைப் பார்த்துவிட்டு பின்வாங்கிச் செல்ல இது வாய்ப்பளிக்கும். எனவே இந்தப் பெட்டி OK மற்றும் Cancel எனும் இரண்டு பொத்தான்களுடன் காணப்படும். OK-வை true எனவும், Cancel-ஐ false எனவும் இது எடுத்துக்கொள்ளும். ஆகவே true-ஆக இருந்தால் என்ன செய்ய வேண்டும், false-ஆக இருந்தால் என்ன செய்ய வேண்டும் என்பதையும் நம்மால் If condition மூலம் வரையறுக்க முடியும்.



User wants to continue!

prompt():

Prompt() box- ஆனது நம்மிடம் ஒரு கேள்வியை எழுப்பி, அதற்கான பதிலை நம்மிடமிருந்து பெற்றுக்கொண்டு, பின்னர் அதற்கேற்ற வகையில் செயல்படும் தன்மை கொண்டது. எனவே இது வெளிப்படுத்துகின்ற பெட்டிக்குள் ஒரு கேள்வியும், ஒரு input box-ம் காணப்படும். கேள்வியாக என்ன வெளிப்பட வேண்டும், input box-ல் என்ன தென்பட வேண்டும் என்பதையெல்லாம், prompt()-க்குள் நாம் வரையறுத்து விடலாம்.



This page says:

Please enter your name :

Aarthi Irudhayaraj

OK Cancel

Hello! Welcome Aarthi Irudhayaraj

9 Exception Handling

try { } என்பது அதன் எல்லைக்குள் உள்ள நிரலில் ஏதேனும் தவறு இருந்தால், அது என்ன தவறு எனும் செய்தியை வெளிப்படுத்தும். catch { } என்பது வெளிப்படுகின்ற செய்தியை பெற்றுக்கொண்டு அதை பயனருக்குத் தெரிவிக்க உதவும்.

கீழ்க்கண்ட உதாரணத்தில் Example1 எனும் பொத்தானின் மீது சொடுக்கும்போது one() எனும் function-ம், Example2 எனும் பொத்தானின் மீது சொடுக்கும்போது two() எனும் function-ம் இயக்கப்படும் விதத்தில் நிரல் எழுதப்பட்டுள்ளது.

```
<html>

  <head>
    <script type="text/javascript">
      <!--
        function one()
        {
          var y = 0;
          try {document.write(x);}
          catch (e) {document.write(e+"</br>");}

          try
          {
```

```
            if (y==0) {throw( "Variable should not hold zero" ); }
          }
          catch (e) {document.write(e+"</br>");}

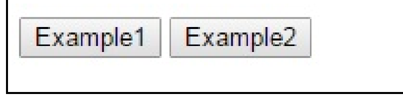
          finally {document.write("Execution Completed" );}
        }
        window.onerror = function (a,b,c)
        {
          alert(a);
          alert(b);
          alert(c);
        }
      //-->
    </script>
  </head>

  <body>
    <form>
      <input type="button" onclick="one()" value="Example1">
      <input type="button" onclick="two()" value="Example2">
    </form>
  </body>

</html>
```

view rawtry_catch.html hosted with by GitHub

இதன் வெளிப்பாடு பின்வருமாறு:



function one()-க்குள் y எனும் variable வரையறுக்கப்பட்டுள்ளது. ஆனால் x -ன் மதிப்பினை வெளியிடுமாறு அடுத்த வரியில் உள்ளது. இந்த x -ஆனது எங்குமே வரையறுக்கப்படாத காரணத்தினால் இது error-ஐ வெளிப்படுத்தி program இயங்குவதை பாதியில் நிறுத்திவிடும். இவ்வாறு நிகழாமல் தடுப்பதற்காகவே x -ன் மதிப்பினை வெளியிடுமாறு உள்ள வரி try{ } -க்குள் கொடுக்கப்பட்டுள்ளது. எனவே ஏதேனும் தவறு நிகழ்ந்தால் கூட அது ஒரு செய்தியாகவே வெளிப்படுமே தவிர, program இயங்குவதற்கு எந்த ஒரு தடையும் இருக்காது.

அடுத்த வரியில் catch (e) எனக் கொடுப்பதன் மூலம் வெளிப்படுகின்ற செய்தியை e எனும் variable மூலம் பெற்றுக்கொண்டு பின்னர் அந்த செய்தியை {document.write(e+”</br>”)} என்பதன் மூலம் வெளிப்படுத்தியுள்ளது. இது பின்வருமாறு.

ReferenceError: x is not defined

இதுபோன்ற Reference error எல்லாம் javascript-ஆல் ஏற்கனவே வரையறுக்கப்பட்ட தவறு வகைகள். சில சமயங்களில் நமது தேவைக்கேற்ப நாமே தவறுகளை வரையறுத்து அதனை சுட்டிக்காட்ட விரும்பினால் throw() பயன்படும்.

அடுத்து கொடுக்கப்பட்டுள்ள try{} block- க்குள் y-ன் மதிப்பு 0 என இருந்தால் "Variable should not hold zero” எனும் error-ஐ வெளிப்படுத்துமாறு throw()-க்குள் கொடுக்கப்பட்டுள்ளது. பின்னர் அந்த error, catch { } மூலம் பெறப்பட்டு

வெளிப்படுத்தப்பட்டுள்ளது. இது பின்வருமாறு.

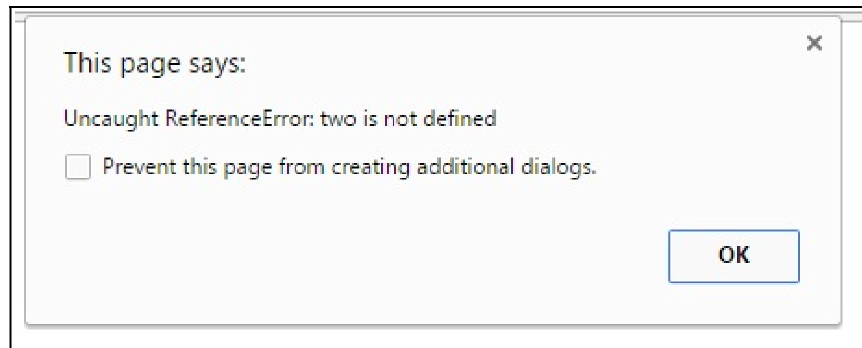
Variable should not hold zero

Finally { } எனும் blocks-க்குள் கொடுக்கப்பட்டுள்ளவை அனைத்தும் errors வந்தாலும் வராவிட்டாலும் கண்டிப்பாக ஒருமுறை execute செய்யப்பட்டு விடும். இது பின்வருமாறு.

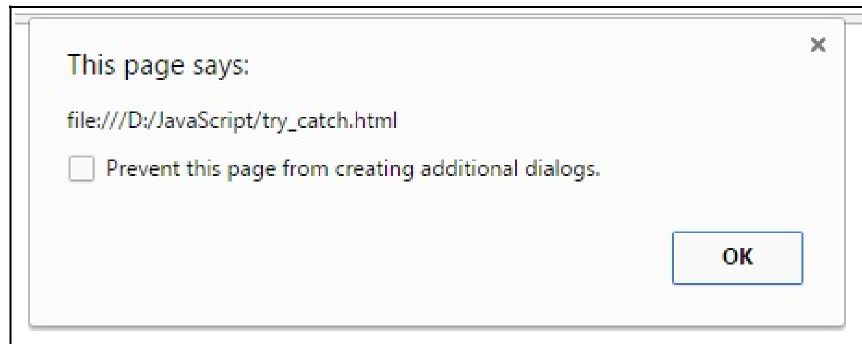
Execution Completed

Function one() – ன் செயல்பாடுகள் முடிந்தவுடன் window.onerror = function (a,b,c) என்று கொடுக்கப்பட்டுள்ளது. அதாவது Example2 பொத்தானின் மீது சொடுக்கும்போது, two() எனும் function, எங்குமே வரையறுக்கப்படாத காரணத்தால் windows- ஆனது error-ஐ வெளிப்படுத்தும். windows.onerror எனும் method, error-ஐ கைப்பற்றி அதனை Error message, url, Line number போன்ற 3 நிலைகளில் சேமிக்கும். இதனை a,b,c போன்ற parameters மூலம் function-க்குள் செலுத்தி alert() மூலமோ அல்லது document.write() மூலமோ வெளிப்படுத்தலாம். இது பின்வருமாறு.

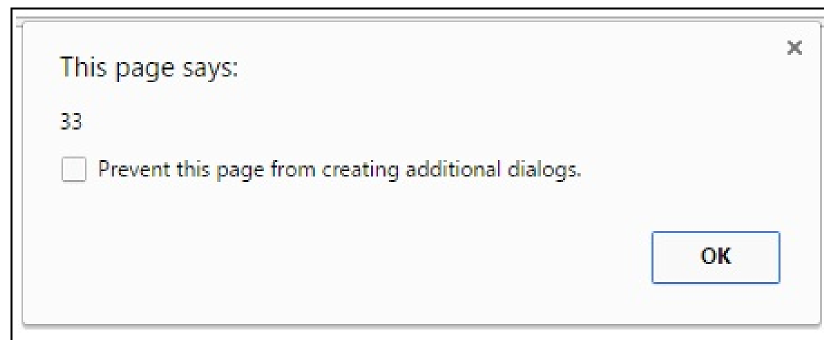
Error Message:



URL:



Line Number:



10 Form Validations, Javascript Objects & Animations

□□□□□□□□□ □□□□□□□□□□□

ஒரு விண்ணப்பப் படிவத்தை நாம் பூர்த்தி செய்துவிட்டு Submit பொத்தானை சொடுக்கினால், உலாவியானது நாம் கொடுத்த விவரங்களை server-க்கு அனுப்புவதற்கு முன்னர், எல்லாம் சரியாக உள்ளதா எனச் சோதிக்கும். ஏதாவது விவரங்களை நாம் கொடுக்கத் தவறியிருந்தாலோ அல்லது தவறுதலாகக் கொடுத்திருந்தாலோ, உலாவியானது ஒரு popup மூலம் அதனை நமக்குத் தெரியப்படுத்தும். சரியான விவரங்களைக் கொடுத்து முழுவதுமாக படிவத்தைப் பூர்த்தி செய்யும்வரை, எந்த ஒரு விவரத்தையும் server-க்கு அனுப்பாது. இதுவே Client side validations எனப்படும். இதனை Javascript எவ்வாறு செய்கிறது என்பதைப் பின்வருமாறு காணலாம்.

கீழ்க்கண்ட உதாரணத்தில் Name, Email, Zip Code எனும் உள்ளீட்டுப் பெட்டிகளும், Country எனும் கீழிறக்கப் பெட்டியும், submit எனும் பொத்தானும் உருவாக்கப்பட்டுள்ளன. இப்போது submit எனும் பொத்தானின் மீது சொடுக்கும்போது, ஒவ்வொரு பெட்டிக்கும் நிகழ வேண்டிய அனைத்து விதமான சோதனைகளும் validate () எனும் ஒரே function-க்குள் கொடுக்கப்பட்டுள்ளன. இந்த அனைத்து சோதனைகளும் வெற்றிகரமாக நடத்தி முடிக்கப்பட்ட பின்னரே இந்த validate()-ஆனது return (true) என்பதனை தெரிவித்து, அடுத்து நடக்க வேண்டிய விஷயத்திற்கே செல்லும். இதில் ஏதேனும் ஒரு சோதனையில் பயனர்கள் அளித்த விவரம் ஒழுங்காக இல்லை என்று கண்டுபிடிக்கப்பட்டுவிட்டால் கூட return(false) எனக் கொடுத்து அடுத்து எதையும் நடக்க விடாது.

```
<html>
  <head>
    <script type="text/javascript">
      <!--
        function validate()
        {
          if( document.myForm.Name.value == "" )
          {
```

```
            alert( "Please provide your name!" );
            document.myForm.Name.focus() ;
            return false;
          }

          var emailID = document.myForm.EMail.value;
          var reg = /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]
{2,4})$/;
          if (reg.test(emailID) == false)
          {
            alert("Please enter correct email ID")
            document.myForm.EMail.focus() ;
            return false;
          }

          if( document.myForm.Zip.value == "" ||
          isNaN( document.myForm.Zip.value ) ||
          document.myForm.Zip.value.length != 5 )
          {
            alert( "Please provide a zip in the format #####." );






            document.myForm.Zip.focus() ;
            return false;
          }
```

```
         if( document.myForm.Country.value == "-1" )
         {
           alert( "Please provide your country!" );
           return false;
         }

         return( true );
       }
     //-->

   </script>
 </head>

 <body>
   <form action="/cgi-bin/test.cgi" name="myForm"
onsubmit="return(validate());">
     <table cellspacing="2" cellpadding="2" border="1">

       <tr>
         <td align="right">Name</td>
         <td><input type="text" name="Name" /></td>
       </tr>

       <tr>
         <td align="right">EMail</td>
         <td><input type="text" name="EMail" /></td>
```

```html
        </tr>

        <tr>
          <td align="right">Zip Code</td>
          <td><input type="text" name="Zip" /></td>
        </tr>

        <tr>
          <td align="right">Country</td>
          <td>
            <select name="Country">
              <option value="-1" selected>[choose yours]</option>
              <option value="1">USA</option>
              <option value="2">UK</option>
              <option value="3">INDIA</option>
            </select>
          </td>
        </tr>

        <tr>
          <td align="right"></td>
          <td><input type="submit" value="Submit" /></td>
        </tr>

      </table>
    </form>
  </body>

</html>
```

view rawValidations.html hosted with by GitHub

Name எனும் பெயர் கொண்ட உள்ளீட்டுப் பெட்டியின் மதிப்பு காலியாக இருந்தால், "Please provide your name!" எனும் எச்சரிப்பு செய்தி வருமாறு நிரல் கொடுக்கப்பட்டுள்ளது. அதைத் தொடர்ந்து Name எனும் பெட்டிக்குள் cursor செல்லுமாறு அமைக்க focus பயன்படுத்தப்பட்டுள்ளது. இவ்வாறே ஒவ்வொரு பெட்டிக்கும் அது பூர்த்தி செய்யப்பட்டுள்ளதா என்பதற்கான சோதனைகளை நாம் செய்து கொள்ளலாம்.

Country எனும் பெயர் கொண்ட கீழிறக்கப் பெட்டியில் 3 நாடுகளின் பெயர்கள் உள்ளது என வைத்துக்கொண்டால் அவை முறையே 0,1,2 என அழைக்கப்படும். எனவே விவரங்கள் எதுவும் தேர்வு செய்யப்படவில்லையா எனக் கண்டுபிடிக்க country-ன் மதிப்பு -1க்குச் சமமாக உள்ளதா எனச் சோதித்தால் போதும். பயனர் dropdown box-ல் எந்த ஒரு நாட்டையும் தேர்வு செய்யவில்லை என்றே அர்த்தம். இவ்வாறே ஒரு கீழிறக்கப் பெட்டிக்கான சோதனைகள் நிகழ்த்தப்படுகின்றன.

நாம் கொடுக்கின்ற email-id சரியாக உள்ளதா என்பதை சோதிப்பதற்கு இணையத்தில் ஒரு regular expression இருக்கும். அதை ஒரு variable-ல் சேமித்து, test()எனும் function-ஐப் பயன்படுத்தி சரியான இடத்தில் தான் @ மற்றும்

. எல்லாம் உள்ளானவா என்பதைச் சோதிக்கலாம். அவ்வாறு இல்லையெனில் "Please enter correct email ID” எனும் எச்சரிப்பு செய்தி வருமாறு அமைக்கலாம்.

அடுத்ததாக Zip எனும் பெயர் கொண்ட உள்ளீட்டுப் பெட்டியின் மதிப்பு காலியாக இருந்தாலோ, எண்கள் அல்லாத வேறு ஏதாவது ஒன்றை மதிப்பாக அளித்தாலோ, நாம் கொடுக்கின்ற மதிப்பு 5 இலக்கத்திற்கு சமமாக இல்லை என்றாலோ "Please provide a zip in the format #####.” எனும் எச்சரிப்பு செய்தி வருமாறு கொடுக்கப்பட்டுள்ளது.

11 Object Oriented Programming Concepts

முதலில் Objects என்றால் என்ன? அதன் தேவை என்ன? அவை எவ்வாறு செயல்படுகின்றன? அவை எவ்வாறு நமது வேலையை சுலபமாக்குகின்றன? என்பதையெல்லாம் இப்பகுதியில் சாதாரண உதாரணங்களை வைத்து நாம் புரிந்து கொள்வோம். இதுவே Javascript-ல் உள்ள Objects-ஐக் கற்பதற்கு சுலபமாக அமையும்.

Object:

அதிக அளவிலான விவரங்களை சேமிப்பதற்கு, நிறைய variables-ஐப் பயன்படுத்தாமல், அவை அனைத்தையும் ஒரே ஒரு Object-க்குள் செலுத்தி, அந்த Object-க்கு ஒரு variable-ஐ உருவாக்குவதன் மூலம் ஒற்றை variable-ல் எண்ணற்ற விவரங்களை சேமிக்கும் விதமே Object Oriented Programming ஆகும்.

Methods & Properties:

பொதுவாக Object என்பது அதற்கான பண்புகள் மற்றும் செயல்பாடுகளுடன் ஒரு Class-க்குள் வரையறை செய்யப்படும். உதாரணத்துக்கு human எனும் ஓர் புதிய Object-ஐ அறிமுகப்படுத்த விரும்பினால், அதற்கான பண்புகள் மற்றும் செயல்பாடுகள் அனைத்தும் ஒரு Class-க்குள் வரையறுக்கப்படும்.

Eg: Class human {Colour, Shape, Race, Speak(), write(), anger()}

ஒரு மனிதனின் நிறம், வடிவம், இனம் போன்ற அவனுக்குரிய பண்புகள் அனைத்தும் Properties எனவும், அவன் செய்கின்ற செயல்களான பேசுதல், எழுதுதல், கோபப்படுதல் ஆகியவை அனைத்தும் Methods எனவும் அழைக்கப்படும்.

இப்போது 100 மனிதனுக்குரிய நிறம், வடிவம், இனம் போன்ற விவரங்களை சேமிக்க விரும்பினால், ஒவ்வொரு விவரத்துக்கும் தனித்தனி variables பயன்படுத்தத் தேவையில்லை. ஒரே variable-ல் அனைத்து வகையான விவரங்களையும் சேமித்துக் கொள்ள முடியும். அதாவது a = human.new() எனக் கொடுத்தால் போதுமானது. a.colour=‘Wheat’, a.Shape=’Short’, a.Race=’Indian’, a.Speak(), a.write(), a.anger() என்ற அனைத்து விதமான விஷயங்களையும், நம்மால் இந்த a எனும் variable-ல் சேமித்து விட முடியும். அதாவது a-வைத் தொடர்ந்து ஒரு புள்ளி வைத்துவிட்டு, human-ன் பண்புகள் மற்றும் செயல்பாடுகளின் பெயர்களைக் கொடுத்து பல்வேறு ஒத்த தகவல்களை சேமித்து அணுகலாம்.

Encapsulation:

இந்த a-வானது human-ன் அனைத்து விதமான செயல்பாடுகளையும், அணுகுகின்ற அதிகாரத்தைப் பெற்று செயல்படுவதே ‘Encapsulation’ எனப்படும். அதாவது எங்கெல்லாம் print a எனக் கொடுக்கிறோமோ, அங்கெல்லாம் அதில் சேமிக்கப்பட்டுள்ள அனைத்து மதிப்புகளும் வெளிப்படும். இந்த a- ஆனது Object-ன் மறுவடிவமாக செயல்படுவதால், இது ஒரு சாதாரண variable-ஆக கருதப்படாமல், Instance என்று அழைக்கப்படும். Object-ன் ஒரு Instance-க்கும், அதன் Methods & Properties-க்கும் உள்ள உறவே "Encapsulation” ஆகும்.

Inheritance:

இப்பொழுது kid எனும் மற்றொரு புதிய object- ஐ உருவாக்கப்போகிறோம். இதற்குள் human-ன் பண்புகள் மற்றும் செயல்பாடுகளுடன் சேர்த்து புதிதாகச் சில பண்புகள் மற்றும் செயல்பாடுகளையும் சேர்ப்பதற்கு,

Class kid (import human) {school, class, cry(), shout()} என்று கொடுக்க வேண்டும். Kid-க்கு உருவாக்கப்படும் instance- ஆல் kid & human எனும் இரண்டு objects-ஐயும் அணுக முடியும். இதுவே Inheritance எனப்படும். இவ்வாறு செய்வதற்கு பதிலாக, human-க்குள் சென்று இந்தப் புதிய பண்புகளையும், அதனுடன் இணைத்து விடலாமே என்று நீங்கள் கேட்கக்கூடும். அவ்வாறு நாம் செய்தோமானால், ஒவ்வொரு முறை human-க்கு instance-ஐ உருவாக்கும்போதும், தேவையில்லாத பண்புகளுக்கும் சேர்த்து memory எடுத்துக்கொள்ளும். எனவேதான் Inheritance என்ற ஒன்று பயன்பாட்டிற்கு வந்தது.

Polymorphism:

Print 5+8 எனக் கொடுக்கும்போது, அது 13 எனும் மதிப்பினை வெளிப்படுத்தும். அதுவே Print '5+8' என நாம் கொடுத்தால், அது 58 எனும் மதிப்பினை வெளிப்படுத்தும். அதாவது Print எனும் ஒரே function, வெவ்வேறு விதங்களில் செயல்படுத்தப்படுகின்றன. இதுவே Polumorphism-க்கு ஓர் சிறந்த உதாரணம் ஆகும்.

12 Javascript Objects

Javascript- ல் பின்வரும் இரண்டு வகையான உள்ளன.

User defined objects

நமக்குத் தேவையான விதத்தில், நாமே உருவாக்கிக் கொள்ளும் Object-ஆனது, User defined object எனப்படும்.

new எனும் operator ஒரு புதிய Object- ஐ உருவாக்கப் பயன்படும். கீழ்க்கண்ட உதாரணத்தில் Var records = new object() என்பதன் மூலம் object() எனும் constructor-அனுப்புகின்ற மதிப்பு records எனும் variable-ல் சேமிக்கப்படுகிறது. எனவே இனி records-ம் ஒரு object போன்றே செயல்படும். எனவே தான் இதற்குள் வரையறுக்கப்பட்டுள்ள மதிப்புகள் அனைத்தும் var name, var age என்று இல்லாமல் object-ன் பண்புகள் எவ்வாறு வரையறுக்கப்படவேண்டுமோ அவ்வாறு வரையறுக்கப்பட்டுள்ளது. records-ஐத் தொடர்ந்து ஒரு புள்ளி வைத்து name, age எனும் இரண்டு பண்புகள் உருவாக்கப்பட்டுள்ளன. அதன் தொடர்ச்சியாக சமக்குறியிட்டு அதன் மதிப்புகளும் வரையறுக்கப்பட்டுள்ளன. அதாவது பண்புகளின் Declaration மற்றும் Initialization ஒரே நேரத்தில் நடைபெற்றுள்ளது. பின்னர் document.write(records.name + records.age) எனக் கொடுக்கும்போதெல்லாம், அவை தான் பெற்றுள்ள மதிப்பினை வெளிப்படுத்துவதைக் காணலாம்.

```
<html>

  <head>
    <script type="text/javascript">
      <!--
        var records = new Object();
        records.name = "Nithya";
        records.age  = 29;

        function attn(x, y){
```

```
            this.name = x;
            this.status = y;
          }
       //-->
    </script>
  </head>

  <body>
    <script type="text/javascript">
       document.write(records.name + records.age + "<br>");

       var attendance = new attn("Madhan", "Present");
       document.write(attendance.name + attendance.status + "<br>");
    </script>
  </body>

</html>
```

view rawUser defined objects.html hosted with by GitHub

அடுத்ததாக ஒரு method-ஐ எவ்வாறு வரையறுப்பது என்றும் மேலே கொடுக்கப்பட்டுள்ளது. இதற்கும் new எனும் operator பயன்படும். <head>- க்குள் attn. என்று ஒரு function வரையறுக்கப்பட்டுள்ளது. இது parameters-ஆக ஒருசில மதிப்புகளை பெற்றுக்கொள்ளும் வகையில் விளங்குகிறது. அவ்வாறு பெற்றுக்கொள்ளும் மதிப்புகளை அதற்குள் வரையறுக்கப்பட்டுள்ள name, status எனும் இரண்டு பண்புகளின் மதிப்பாக செலுத்துகிறது. இவை அனைத்தும் this.name, this.status எனக் கொடுக்கப்பட்டுள்ளது. ஏனெனில் function என்ற ஒன்று தன்னிச்சையாக செயல்படக்கூடியது. Method என்பது Object-வுடன் இணைந்து function-ன் மீது செயல்பட்டு ஏதோ ஒன்றை நிகழ்த்தும் தன்மை உடையது. எனவே இது போன்ற methods-ன் பயன்பாட்டிற்காக function-

க்குள் உள்ள அனைத்தும் this. என்று துவங்க வேண்டும்.

<body>-க்குள் attendance எனும் ஒரு object-ஆனது attn. method-க்கு உருவாக்கப்பட்டுள்ளது. இந்த object-ஐ உருவாக்கும்போதே அதன் வழியாக function-க்குள் செலுத்தப்பட வேண்டிய parameters-ம் கொடுக்கப்பட்டுவிட்டன. attendance-ஐத் தொடர்ந்து புள்ளி வைத்து method.-க்குள் வரையறுக்கப்பட்ட பண்புகளைப் பயன்படுத்தி அதன் மதிப்புகளை வெளிப்படுத்துவது இங்கு கொடுக்கப்பட்டுள்ளது.

இவ்வாறே ஒரு user defined object-ஐ வரையறுத்து நாம் பயன்படுத்தலாம்.

Built-In Objects

ஒரு மொழி உருவாக்கப்படும்போதே வரையறுக்கப்படுகின்றவை Built-In Objects எனப்படும். Javascript- ல் Number, Boolean, Strings, Arrays, Date, Math & RegExp போன்ற பல்வேறு வகையான Built-In Objects Objects உள்ளன. இவற்றின் Properties மற்றும் Methods- ன் பயன்பாடுகளைக் கீழே காணலாம்.

```
<html>
  <head>

    <script type="text/javascript">
      <!--
        function a1()
        {

          document.write
(Number.MAX_VALUE+'<br/>'+Number.MIN_VALUE+'<br/>'+Number.POSIT
+Number.NaN+'<br/>'
+Number.prototype+'<br/>'
+Number.constructor);
        }
      //-->
    </script>

  </head>

  <body>

    <form>
```

```
        <input type="button" value="Click Me" onclick="a1();" />
      </form>

   </body>
</html>
```

view rawBuilt in objecets_numbers_properties.html hosted with by GitHub

```
<html>

   <head>
   </head>

   <body>
      <script type="text/javascript">

        var num=28542;
        document.write
(num.toExponential()+'<br/>'+num.toExponential(5)+'<br/>'+num.toFixed(2)+'<

      </script>
   </body>

</html>
```

view rawBuilt in objects_numbers_methods.html hosted with by GitHub

இது Number எனும் Object-ன் Properties மற்றும் Methods எப்படி செயல்படுகின்றன என்பதைக் காட்டுகிறது. இதை உங்களாலேயே புரிந்து கொள்ள முடியும். இவ்வாறே Boolean, Strings, Arrays, Date, Math & RegExp போன்ற

பலவற்றுக்கும் பல்வேறு வகையான Properties மற்றும் Methods உள்ளன. இவற்றை இணையத்தின் துணை மூலம் தெரிந்து கொள்ளவும்.

13 Animation

Animation என்பது செயல்படக் கூடிய அல்லது இயங்கக்கூடிய விதத்தில் உலாவிகளில் படங்களை உருவாக்குகின்ற முறை ஆகும். அதாவது நிலையாக இருக்கும் ஒரு படமானது இயங்க ஆரம்பித்துவிட்டால் அதுவே animation எனப்படும்.

கீழ்க்கண்ட உதாரணத்தில் "Move 1 inch" எனும் பொத்தானின் மீது சொடுக்கும்போது திரையில் உள்ள படமானது வலப்பக்கமாக ஒரு இன்ச் நகரும் வகையிலும், Move right எனும் பொத்தானின் மீது சொடுக்கும்போது அதே படமானது முழுவதுமாக வலதுபுறத்தில் நகர்ந்து கொண்டே செல்லும் வகையிலும் நிரல்கள் எழுதப்பட்டுள்ளன. Stop என்பது நகர்ந்து கொண்டே செல்லும் படத்தை நிறுத்த உதவும் ஒரு பொத்தான் ஆகும்.

```
<html>
  <head>
    <script type="text/javascript">
      <!--
        var imgObj = null;

        function init(){
          imgObj = document.getElementById('myImage');
          imgObj.style.position= 'relative';
          imgObj.style.left = '0px';
        }

        function moveinch(){
```

```
            imgObj.style.left = parseInt(imgObj.style.left) + 10 + 'px';
          }

          function moveRight(){
            imgObj.style.left = parseInt(imgObj.style.left) + 10 + 'px';
            animate = setTimeout(moveRight,20); // call moveRight in 20msec
          }

          function stop(){
            clearTimeout(animate);
            imgObj.style.left = '0px';
          }

          window.onload =init;
        //-->

      </script>
   </head>

   <body>
      <form action="/cgi-bin/test.cgi" name="myForm"
onsubmit="return(validate());">
        <img id="myImage" src="D:\JavaScript\img1.png" />
        <p>Click button below to move the image to right</p>
        <input type="button" value="Move 1 inch" onclick="moveinch();" />
        <input type="button" value="Move right" onclick="moveRight();" />
        <input type="button" value="Stop" onclick="stop();" />
      </form>
   </body>
</html>
```

14 jQuery - □□□ □□□□□□□□

jQuery என்பது Javascript-ஐ மையமாக வைத்து உருவாக்கப்பட்ட ஒரு framework ஆகும். வரிவரியாக நிரல்களை எழுதி Javascript செய்யும் ஒருசில வேலைகளை jQuery- ஆனது சுலபமாகச் செய்துவிடும். அதாவது ஒரு வேலையை செய்வதற்கு பக்கம் பக்கமாக javascript-ல் நிரல்கள் தேவைப்படின், அவை அனைத்தும் jQuery-ன் ஒரு method-க்குள் அடங்கிவிடும். எனவே அந்த method-ஐ மட்டும் அழைத்து இயக்கினால் போதுமானது. சுருங்க நிரல் அடித்து விரிவான வேலைகளை செய்து முடிக்கும் சிறப்பினை jQuery பெறுகிறது. இது வலைத்தளப் பக்கங்களின் html மொழியுடன் தொடர்புகொண்டு அதன்மீது methods-ஐ இயக்குவதன் மூலம் பயனர்கள் விரும்பும் மாற்றங்களை நிகழ்த்துகிறது. இதுவே jQuery போன்ற frameworks-ன் சிறப்பியல்பு ஆகும்.

SQL query-ஐப் பயன்படுத்தி எப்படி நம்மால் நமக்கு வேண்டிய தகவல்களை database-ல் இருந்து பெற்றுக்கொள்ள முடியுமோ, அவ்வாறே jQuery-யைப் பயன்படுத்தி, நமக்கு வேண்டிய html tags-ஐ வலைத்தளப் பக்கங்களிலிருந்து நம்மால் பெற்றுக்கொள்ள முடியும். எனவேதான் இதற்கு jQuery எனப் பெயர்வந்தது. அவ்வாறு எடுக்கப்பட்ட tags-ஐ நமக்கு விரும்பிய விதத்தில் மாற்றவோ இயக்கவோ செய்யலாம்.

எடுத்துக்காட்டுக்கு ஒரு பக்கத்தின் <p> tags-க்குள் வரையறுக்கப்பட்ட வரிகளை மட்டும் நீல நிற எழுத்துக்களில் மாற்ற விரும்பினால் அதற்கான நிரல்கள் பின்வருமாறு அமையும்.

```
<html>

  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script>
    jQuery(document).ready(function(){
      jQuery("button").click(function(){

        var x = jQuery("p");
        x.css("color","blue");

      });
    });
    </script>
  </head>

  <body>
      <p>Getting an identity is important</p>
      <button>Desired Changes</button>
  </body>

</html>
```

view raw01_jquery_sample.html hosted with by **GitHub**

:

Getting an identity is important

jQuery :

Getting an identity is important --

-

-

-

-

-

:

jQuery-யானது எப்போதும் <head>க்குள் உள்ள <script> tag-க்குள் எழுதப்பட்டிருக்கும். இதன் அம்சங்களை நமது program-ல் பயன்படுத்தத்

துவங்குவதற்கு முன்னர், jQuery CDN-ஐ இந்த <script> tag மூலம் முதலில் இணைக்க வேண்டும். Google, Microsoft போன்றவை இத்தகைய CDNs (Content Delivery Network)-ஐ வழங்குகின்றன. இங்கு பயன்படுத்தியுள்ள ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js என்பது google-வழங்குகின்ற ஒன்றாகும்.

மேற்கண்ட எடுத்துக்காட்டில் <p> tags- க்குள் Getting an identity is important எனும் வரி கொடுக்கப்பட்டுள்ளது. jQuery("p”) என்பது <p> tags- க்குள் வரையறுக்கப்பட்டுள்ளவற்றை எடுத்து x எனும் variable-ல் சேமித்து வைத்துக்கொள்கிறது. பின்னர் css() எனும் method மூலம் என்ன செய்ய வேண்டும் என்பது வலியுறுத்தப்படுகிறது. அதாவது x-ல் சேமிக்கப் பட்டுள்ளவற்றை நீல நிறத்தில் மாற்றுவதற்கு x.css("color”,”blue”) என்று கொடுக்கப்பட்டுள்ளது. இங்கு நாம் நிகழ்த்த விரும்பும் மாற்றம் arguments-ஆக வழங்கப்பட்டுள்ளது.

நான் இவை அனைத்தையும் ஒரு function() {—} -க்குள் எழுதியுள்ளேன். பின்னர் Desired Changes எனும் பொத்தானை சொடுக்கும்போது இந்த function இயக்கப்படும் வகையில் அதனை அமைத்துள்ளேன். அதாவது jQuery("button”).click()-க்குள் இந்த function-ஐ வைத்துள்ளேன். ஏனெனில் அப்போதுதான் உங்களால் சாதாரணமாக எப்படி இருந்தது, jQuery-என்ன மாற்றம் செய்தது எனும் வித்தியாசத்தை புரிந்து கொள்ள முடியும். வலைத்தளப் பக்கமானது முழுவதுமாக load செய்யப்பட்ட பின்னர் இவையெல்லாம் செயல்படத் துவங்கும் வகையில், இவை அனைத்தும் jQuery(document).ready()-க்குள் கொடுக்கப்பட்டுள்ளதை கவனிக்கவும். ஏனெனில் jQuery ஆனது முழுக்க முழுக்க html tags-ஐ அடிப்படையாகக் கொண்டு இயங்குவதால், jQuery செயல்படத் துவங்குவதற்கு முன்னர், இந்த html பக்கங்களானது ஒழுங்கான முறையான வடிவமைப்புகளுடன் load செய்யப்படுவது அவசியம். எனவேதான் இவை அனைத்தும் document.ready()-க்குள் கொடுக்கப்பட்டுள்ளன. அவ்வளவுதான்! இவையே அடிப்படையான விஷயங்கள்.

மேலும் jQuery() என்பது $ எனும் குறியீட்டின் மூலமும் குறிக்கப்படும். அதாவது jQuery(“p”) என்பதை $("p”) என்றும் குறிப்பிடலாம். அவ்வாறே jQuery("p”) மூலம் பெறப்படும் tags-ஐ ஒரு variable-ல் சேமித்து, பின்னர் அதன்மீது .css() – ஐப் பயன்படுத்துவதற்குப் பதிலாக $("p”).css ("color”,”blue”) என்றும் குறிப்பிடலாம். இனிவரும் எடுத்துக்காட்டுகளில் இதுபோன்ற அமைப்புமுறைகளே

பயன்படுத்தப்பட்டிருக்கும்.

15 jQuery – CSS – Animations

Jquery CSS-□ □□□□□□□□ □□□□□

CSS என்பது HTML மூலம் உருவாக்கப்படும் பக்கங்களை இன்னும் அழகு படுத்த உதவும். அதாவது எழுத்துக்களின் வகைகள், நிறங்கள், பின்புற வண்ணங்கள் போன்ற பல்வேறு வகையான அழகு சார்ந்த விஷயங்களை ஒருசேர தொகுத்துக் கொடுக்க இந்த css உதவும். "அழகிய பக்கங்களின் ஊற்று" என்பதே "Cascading style sheets" என்பதன் தமிழாக்கம் ஆகும். இதைப் பற்றி முழுமையாகத் தெரிந்து கொள்ள www.kaniyam.com/learn-css-in-tamil-ebook/ எனும் முகவரியை பார்க்கவும். இதுபோன்ற css பண்புகளின் மீது jQuery-யை எவ்வாறு பயன்படுத்தலாம் என்று இதில் பார்க்கலாம்.

கீழ்க்கண்ட எடுத்துக்காட்டில் css பண்புகள் அனைத்தும் <style> எனும் tag-க்குள் வரையறுக்கப்பட்டுள்ளன. இதுவே css-ஐ குறிப்பிடும் முறை ஆகும். jquery object-ன் மீது பயன்படுத்தப்படும் .css() என்பது குறிப்பிட்ட பண்பினை வெளிப்படுத்த உதவும். அவ்வாறே .css()-க்குள் கொடுக்கப்படும் மதிப்புகளுக்கு ஏற்ப அந்த element-ஐ மாற்றி அமைக்கவும் செய்யும். அதாவது 'Getting value' என்பதன் மீது சொடுக்கும்போது p1-ன் பின்புற வண்ணம் வெளிப்படுமாறும், 'Setting value' -ன் மீது சொடுக்கும்போது p1-ன் எழுத்துக்களின் வடிவம் மாறுமாறும் இங்கே நிரல்கள் கொடுக்கப்பட்டுள்ளன.

<html>
<head>
<style type="text/css">
#p1
{
color: blue;
background-color: pink;
font-weight: bold;

```
    }
  </style>

  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
  </script>

  <script>
  $(document).ready(function(){

    $("#1").click(function(){
      alert ($("#p1").css("background-color"));
    });

    $("#2").click(function(){
      $("#p1").css("font","italic 14pt sans-serif");
    });

  });
  </script>

  </head>

  <body>
      <p id='p1'>Getting an identity is important</p>
      <button id='1'>Getting value</button>
      <button id='2'>Setting value</button>
  </body>
```

</html>

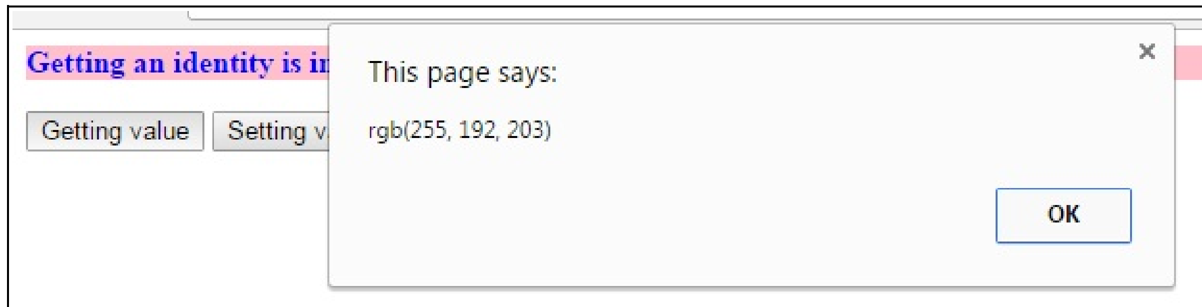
view raw05_jquery_css.html hosted with by GitHub

□□□□□□□ □□□□□□□□□□□:

Getting an identity is important

Getting value Setting value

Getting value-□□ □□□□ □□□□□□□□□□□□□ □□□□□□□□□□□□:



Setting value-□□ □□□□□ □□□□□□□□□□□□□ □□□□□□□□:

Getting an identity is important

Getting value Setting value

Jquery elements-□□ □□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□ □□□□□□

வலைத்தளப் பக்கங்களில் இயக்கங்களை உருவாக்குவதற்கும், அவ்வியக்கங்களை கண்கூடாகக் காண்பதற்கும் உதவும் jquery methods-ஐ இங்கு காணலாம். கீழ்க்கண்ட எடுத்துக்காட்டில் ஒருசில அடிப்படையான இயக்கங்களும், அதற்கான விளக்கங்களும் கொடுக்கப்பட்டுள்ளன.

<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("#p1").hide(600);
$("#p3").toggle("slow");
$("#f1").slideUp("slow");
$("#dv1").fadeOut("slow");
$("#dv2").fadeTo(400,0.5);
$("#dv3").animate({"font-size" : "24pt"}, "slow");

```
        });

    });
    </script>
  </head>

  <body>
        <p id='p1'>Getting an identity is important</p>
        <p id='p2'>Doing the right things at right time</p>
        <p id='p3'>Best things will always comes to you</p>
        <p id="p4">Big data is the emerging trend</p>
        <p id="p5">Data Science is different from Big data</p>
        <div id="dv1">You can feel the freedom everywhere</div>
        <div id='dv2'>win-win situation is not always possible in family
world</div>
        <div id="dv3">Come out of the shell</div>
        <form id='f1' action="">
          <input id="r1" type="radio" name="gender" value="female">
Female<br>
          <input id="r2" type="radio" name="gender" value="male"> Male<br>

        </form>
        <button>Desired Changes</button>
  </body>

</html>
```

view raw06_jquery_animations.css hosted with by GitHub

□□□□□□□□□□□ □□□□□□□□□:

hide() -ஆனது p1-எனும் id-ஐக் கொண்ட வரியை மறைக்கிறது. இதற்குள் கொடுக்கப்பட்ட 600 என்பது மறைத்தல் நடைபெறும் விதத்தை மெதுவாகக் காட்ட உதவும்.

toggle() என்பது மறைக்கப்பட்ட element-ன் அடிப்படையில் முன்னும் பின்னும் நகர்வதற்கு பயன்படும். இங்கு p3-ன் மீது பயன்படுத்தப்பட்ட toggle() -ஆனது p1 மறைக்கப்பட்ட பின்னர் அதனிடத்தை அடைகிறது.

slideUp() என்பது f1-எனும் படிவத்தை மேற்புறமாக நகர்த்தி மறைக்கிறது.

fadeOut() என்பது dv1-எனும் element-ஐ கொஞ்சம் கொஞ்சமாக மங்கடித்து மறைக்கிறது.

fadeTo(400,0.5) என்பது dv2-எனும் element-ஐ குறிப்பிட்ட அளவில் மங்கடித்து மங்கிய நிலையில் அமைக்கிறது.

animate() என்பது நாமாக ஒருசில இயக்கங்களை வரையறுக்க உதவும் method ஆகும். இங்கு animate({"font-size" : "24pt"}, "slow") என்பது dv3 எழுத்துக்களின் அளவினை பெரிதுபடுத்த உதவியுள்ளது. அவ்வாறு பெரிது படுத்தும் நிகழ்வானது மெதுவாக நிகழ வேண்டும் என்பதை slow உணர்த்துகிறது.

□□□□□□□ □□□□□□□□□□□:

Getting an identity is important

Doing the right things at right time

Best things will always comes to you

Big data is the emerging trend

Data Science is different from Big data

You can feel the freedom everywhere
win-win situation is not always possible in family world
Come out of the shell
Female
Male

Desired Changes

jQuery □□□□□□□□□□□□ □□□□□□□□:

Doing the right things at right time

Big data is the emerging trend

Data Science is different from Big data

win-win situation is not always possible in family world

Come out of the shell

Desired Changes

16 jQuery- □□□□□□□□□ □□□□□□□□□□□ □□□□□□□□□ □□□□□□□□□

jQuery மூலம் வலைத்தளப் பக்கங்களில் உள்ளவற்றை மாற்றி அமைக்க முடியும். படங்கள், படிவங்கள், செய்திகள் போன்ற அனைத்து விதமான விஷயங்களையும் jQuery-மூலம் அணுகவோ மாற்றி அமைக்கவோ முடியும். இவை ஒவ்வொன்றும் விவரமாகக் கீழே கொடுக்கப்பட்டுள்ளன.

attr( ) □□□□□ □□□□□□□□ □□□□□□□□□□□□□□

jQuery மூலம் தேர்ந்தெடுக்கப்பட்ட ஒரு object-ஐ attr() எனும் பண்பின் மூலம் நாம் விரும்பிய வகையில் மாற்றி அமைக்க முடியும். கீழ்க்கண்ட எடுத்துக்காட்டில் "Modify Image" பொத்தானை சொடுக்கும்போது வேறொரு படம் திரையில் வெளிப்படுமாறும், "Modify sizing" -ஐ சொடுக்கும்போது அதன் நீள அகலம் மாறும் வகையிலும், "Remove sizing" –ஐ சொடுக்கும்போது அந்த மாற்றங்கள் நீங்கும் வகையிலும் நிரல்கள் கொடுக்கப்பட்டுள்ளன.

```
<html>

  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script>
    $(document).ready(function(){

      $("#1").click(function(){
        $("#flower").attr("src","two.jpg");
      });

      $("#2").click(function(){
        $("#flower").attr( {"src":"two.jpg","height":"100","width":"500"} );
      });
      $("#3").click(function(){
        $("#flower").removeAttr("width height");
      });
    });
```

```
    </script>
  </head>

  <body>
    <button id='1'>Modify Image</button>
    <button id='2'>Modify sizing</button>
    <button id='3'>Remove sizing</button>
    <img id='flower' src="one.jpg">
  </body>

</html>
```

view raw04_1_jquery_modify_images.html hosted with by GitHub

இங்கு 1 எனும் பெயர் கொண்ட "Modify Image” பொத்தானை சொடுக்கும்போது flower எனும் பெயர் கொண்ட <img> tag-ன் மூலமாக வேறு ஒரு படம் கொடுக்கப்பட்டுள்ளது. இதற்கு attr() function பயன்பட்டுள்ளது. எனவே Modify Image என்பதைச் சொடுக்கும்போது ஏற்கனவே உள்ள படம் மறைந்து நாம் கொடுத்துள்ள two.jpg எனும் படம் தென்படும்.

அவ்வாறே 2 எனும் பெயர் கொண்ட "Modify sizing” பொத்தானை சொடுக்கும்போது திரையில் உள்ள படத்தின் நீள அகலம் மாறுமாறு attr() மூலம் மீண்டும் மதிப்புகள் கொடுக்கப்பட்டுள்ளன.

removeAttr("width height”) என்பது படத்தின் நீளம் மற்றும் அகலத்தில் செய்யப்பட்ட மாற்றங்களை நீக்கி அதனை பழைய நிலைக்குக் கொண்டுவரும். 3 எனும் பெயர் கொண்ட "Remove sizing” பொத்தானை சொடுக்கும்போது, இந்த மாற்றம் நிகழும் வகையில் நிரல் கொடுக்கப்பட்டுள்ளது.

:

Modify Image Modify sizing Remove sizing



Modify Image- :

Modify Image Modify sizing Remove sizing



Modify sizing- :



text( ) □□□□□□ html( ) □□□□□ content-□ □□□□□□□□□□□□□□

jQuery object- ன் மீது .text() என்று கொடுத்தால் அதன் text content-ம், .html() என்று கொடுத்தால் அதன் html content-ம் வெளிப்படும். அவ்வாறே .text(" ******* ”)-க்குள் நாம் கொடுக்கும் விஷயத்தை அந்த jquery object-ன் மீது மாற்றி அமைக்கும். அதையே .html(" ******* ”)-க்குள் கொடுத்தால், அது html மொழியின் அர்த்தங்களைப் புரிந்துகொண்டு அதன்படி மாற்றி அமைக்கும்.

கீழ்க்கண்ட எடுத்துக்காட்டில் All the glitters are not gold -famous proverb என்பது text( )-க்குள் கொடுக்கப்படும் போது அது அப்படியே திரையில் வெளிப்படுவதையும், அதையே html( )-க்குள் கொடுத்தால் - ன் அர்த்தத்தைப் புரிந்துகொண்டு கொடுக்கப்பட்ட வரியை பெரிய எழுத்துக்களில் வெளிப்படுத்துவதையும் காணலாம்.

```
<html>

  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script>
    $(document).ready(function(){

      $("#1").click(function(){
        alert ( $("#01").text() );
        alert ( $("#01").html() );
      });

      $("#2").click(function(){
```

```
            $("#01").text("<b>All the glitters are not gold </b> -famous proverb");
         });

         $("#3").click(function(){
            $("#01").html("<b>All the glitters are not gold </b> -famous proverb");
         });

      });
      </script>
   </head>

   <body>
         <p id='01'> <b>All the world is a stage </b> -famous quote</p>
         <button id='1'>Reading content - text & html</button>
         <button id='2'>Modify content - text()</button>
         <button id='3'>Modify content - html()</button>
   </body>

</html>
```

view raw04_2_jquery_modify_text vs html.html hosted with by **GitHub**

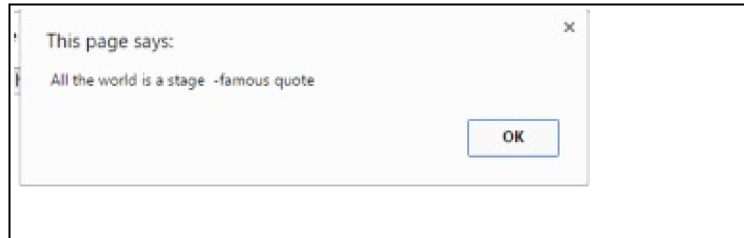
:

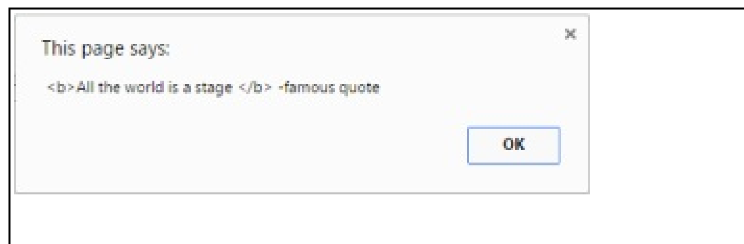
All the world is a stage -famous quote

Reading content - text & html | Modify content - text() | Modify content - html()

Reading content – text & html -

:





Modify Content – text() - **:**



Modify Content – html() - :



Offset() □□□□□ content-□ □□□□ □□□□□□□□□□

Offset() எனும் method-ஆனது jQuery object-ஆல் குறிக்கப்படும் element வலைத்தளப் பக்கத்தில் சரியாக எந்த இடத்தில் உள்ளது என்பதை pixel வடிவில் வெளிப்படுத்தும். offset({****, ***})- க்குள் கொடுக்கப்படும் மதிப்புகளுக்கு ஏற்ப அந்த element-ஐ இடம் மாற்றியும் அமைக்கும்.

கீழ்க்கண்ட எடுத்துக்காட்டில் $("#dv1”)- ஆல் குறிக்கப்படும் element இடது பக்கத்திலிருந்து எவ்வளவு தொலைவில் உள்ளது என்பதையும், மேற்புறத்திலிருந்து எவ்வளவு தொலைவில் உள்ளது என்பதையும் pixel வடிவில் வெளிப்படுத்தும். பின்னர் அந்த element-ஐ {"left”:50, "top”:100} என்பதற்கேற்ப இடம் மாற்றி அமைக்கும்.

```
<html>

  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script>
    $(document).ready(function(){

      $("#1").click(function(){
        alert ($("#dv1").offset().left + "," + $("#dv1").offset().top);
      });

      $("#2").click(function(){
        $("#dv1").offset({"left":50, "top":100});
```

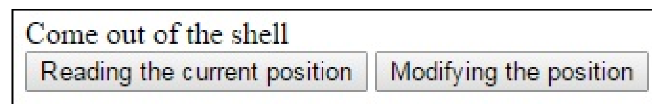
```
        });
    });
    </script>
  </head>

  <body>
      <div id="dv1">Come out of the shell</div>
      <button id='1'>Reading the current position</button>
      <button id='2'>Modifying the position</button>
  </body>

</html>
```

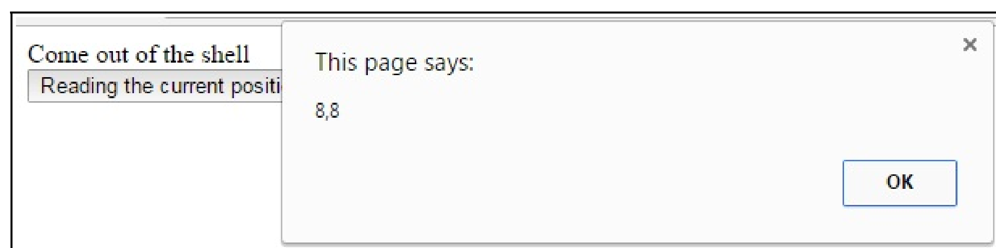
[view raw04_3_jquery_modify_positions.html](#) hosted with by **GitHub**

:



Reading the current position-

:



Modifying the position- **:**

Reading the current position | Modifying the position

Come out of the shell

val() □□□□□ □□□□□□□□□ □□□□□□□□□□

val() எனும் method-ஆனது விண்ணப்பப் படிவங்களில் பூர்த்தி செய்யப்பட்ட மதிப்புகளை வெளிப்படுத்த உதவுகிறது. அவ்வாறே val(" ")-க்குள் உள்ள மதிப்புகளை விண்ணப்பப் படிவங்களின் மதிப்புகளாக படிவப் பெட்டிகளுக்குள் செலுத்தவும் செய்கிறது.

கீழ்க்கண்ட எடுத்துக்காட்டில் 1 எனும் பொத்தானை சொடுக்கும்போது படிவங்களில் நாம் பூர்த்தி செய்த மதிப்புகள் வெளிப்படுமாறும், 2 எனும் பொத்தானை சொடுக்கும்போது India, Green, male ஆகிய மதிப்புகள் படிவங்களின் மதிப்புகளாக மாற்றி அமைக்கப்படுமாறும் நிரல்கள் கொடுக்கப்பட்டுள்ளன.

```
<html>

  <head>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script>
    $(document).ready(function(){

      $("#1").click(function(){
        alert ( $("#a1").val() );
        alert ( $("#a2").val() );
        alert ( $("[name=gender]:checked").val() );
      });

      $("#2").click(function(){
        $("#a1").val("India");
```

```
        $("#a2").val("Green");
        $("input[name=gender]").val(["male"]);
      });



    });
    </script>
  </head>

  <body>
      <form action="">
        <input id="a1" type="text" name="address" value="Enter Address..."/>
        <select id="a2" name="color" size="3">
          <option>Red</option>
          <option>Green</option>
          <option>Blue</option>
        </select>
        <input id="r1" type="radio" name="gender" value="female"> Female
        <input id="r2" type="radio" name="gender" value="male"> Male
      </form>
      <button id='1'>Reading the values</button>
      <button id='2'>Modifying the values</button>
  </body>

</html>
```

:

Red
Green
Blue

Enter Address... Female Male

Reading the values Modifying the values

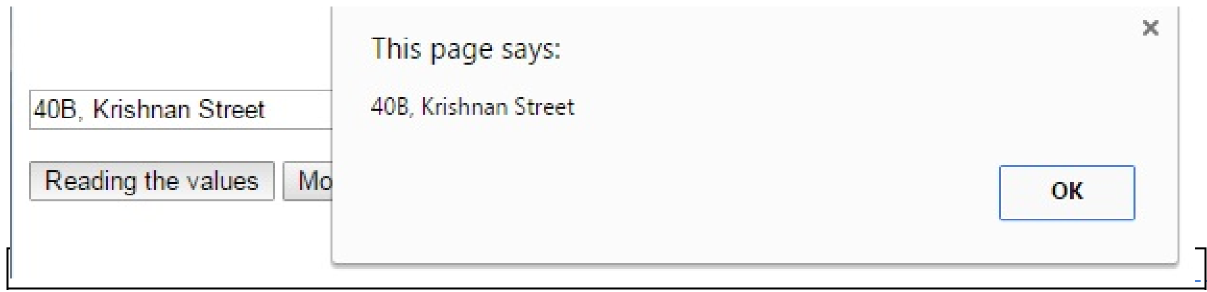
Red
Green
Blue

40B, Krishnan Street Female Male

Reading the values Modifying the values

Reading the values-

:



இதைத் தொடர்ந்து அடுத்தடுத்த மதிப்புகள் வெளிப்படும்.

Modifying the values- :

data() □□□□□□ comments-□□ □□□□□□□ □□□□□□□ □□□□□□□□□

data(" ")-க்குள் கொடுக்கப்படும் வரிகள் ஒரு popup பெட்டி மூலம் comment-ஆகத் தென்படும். கீழ்க்கண்ட எடுத்துக்காட்டில் p1 எனும் id-ஐக் கொண்ட paragraph மீது சொடுக்கும்போது comments வெளிப்படுமாறு நிரல்கள் கொடுக்கப்பட்டுள்ளன. இந்த comments-ஆனது மறைமுகமாக p1- வுடன் இணைந்திருக்கும். p1-ன் மீது சொடுக்கினால் மட்டுமே அது தென்படும். ‘Remove Comment’ -ன் மீது சொடுக்கும்போது p1- வுடன் மறைமுகமாக இணைந்திருக்கும் comments நீங்குமாறு நிரல்கள் எழுதப்பட்டுள்ளன.

<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
var a = {
note1 : "Ahhh gaaga gaagaa",
note2 : "Adi sakka",
toString : function ()
{

```
            return this.note1 + "\n" + this.note2;
          }
        };

      $("#p1").data("notes",a);

      $("#p1").click(function(){
        var b = $("#p1").data("notes");
        alert ( b );
      });

      $("#1").click(function(){
        $("#p1").removeData("notes");
      });

    });
    </script>
  </head>

  <body>
    <p id='p1'>Finally the girl the one who is doing everything from the
begining</p>
    <button id='1'>Remove comment</button>
  </body>

</html>
```

view raw04_5_jquery_modify_comments.html hosted with by GitHub

:

Finally the girl the one who is doing everything from the begining

Remove comment

Finally the girl

comment:

Finally the girl the one w

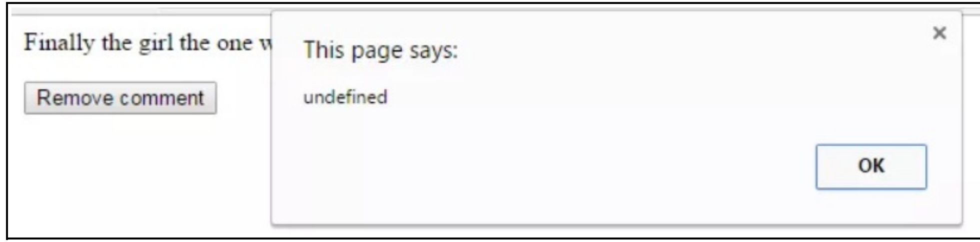
Remove comment

This page says:

Ahhh gaaga gaagaa
Adi sakka

OK

Remove comment பொத்தானை சொடுக்கியபின் Finally the girl எனத் துவங்கும் வரியின் மீது சொடுக்கினால் தென்படுவது:



□□□□□□□□□□ □□□□□□□□□□□ content-□ □□□□□□□□□□□□□□□

வலைத்தளப் பக்கங்களில் உள்ள வரிகளை மேலும் கீழுமாக இடம் மாற்றம் செய்தல், புதிய வரிகளை இணைத்தல், ஏற்கனவே உள்ள வரிகளை நகலெடுத்து மீண்டும் இணைத்தல், ஒருசில வரிகளை நீக்குதல் போன்ற பல்வேறு வேலைகளை செய்வதற்கு உதவும் jQuery methods-ஐ இந்தப் பகுதியில் பார்க்கலாம்.

<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("#p1").before($("#p2"));
$(" All the <i>leaves</i> are falling down ").insertBefore($("#p3"));
$("#p3").clone().attr("id","abcd_copy").appendTo($("#dv1"));
$("#p4").wrap($("#p5"));
$("#dv2").replaceWith($("#p5"));
$("#f1").remove();
$($.parseHTML("<i>All the leaves</i> are falling down")

```
).insertAfter($("#dv3"));
        });

    });
    </script>
  </head>

  <body>
        <p id='p1'>Getting an identity is important</p>
        <p id='p2'>Doing the right things at right time</p>
        <p id='p3'>Best things will always comes to you</p>
        <p id="p4">Big data is the emerging trend</p>
        <p id="p5">Data Science is different from Big data</p>
        <div id="dv1">You can feel the freedom everywhere</div>
        <div id='dv2'>win-win situation is not always possible in family
world</div>
        <div id="dv3">Come out of the shell</div>
        <form id='f1' action="">
            <input id="r1" type="radio" name="gender" value="female">
Female<br>
            <input id="r2" type="radio" name="gender" value="male"> Male<br>
        </form>
        <button>Desired Changes</button>
  </body>

</html>
```

view raw04_6_jquery_modify_webcontent.html hosted with by GitHub

:

before() -ஆனது p2-எனும் id-ஐக் கொண்ட வரியை p1-க்கு முன்னால் அமைக்கிறது.

insertBefore() -ஆனது All the *leaves* are falling down எனும் புதிய வரியை p3-க்கு முன்னால் இணைக்கிறது. இந்தப் புதிய வரியானது $(" ") -க்குள் சாதாரண html tag-ஐ கொடுப்பதன் மூலம் உருவாக்கப்படும்.

Clone() -ஆனது p3 id-ஐக் கொண்ட வரியை நகலெடுத்து உடனடியாக .attr() மூலம் அதன் id-ஐ மட்டும் மாற்றி விடுகிறது. ஏனெனில் ஒரே id-ல் இரண்டு elements இருப்பது பல குழப்பங்களை விளைவிக்கும். பின்னர் நகலெடுக்கப்பட்ட வரியை dv1-க்குப் பின்னால் appendTo() மூலம் இணைக்கிறது.

wrap() என்பது p4, p5 எனும் இரண்டையும் ஒன்றாக இணைக்கிறது.

replacewith() என்பது dv2-ஐ நீக்கிவிட்டு p5-ஐ அந்த இடத்தில் அமைக்கிறது. அவ்வாறு அமைக்கப்படும்போது p5-வுடன் சேர்ந்து p4-ம் இடம்பெறுவதை கவனிக்கவும். ஏனெனில் இதற்கு முந்தைய வரியில் p4,p5 இரண்டும் wrap செய்யப்பட்டதே இதற்குக் காரணம் ஆகும்.

remove() -ஆனது f1-எனும் id-ஐக் கொண்ட படிவத்தை நீக்குகிறது.

$.parseHTML(" ") –ஆனது இதற்குள் கொடுக்கப்பட்டுள்ள html-மொழியினைப் புரிந்துகொண்டு அதற்கேற்ற வடிவில் வெளிப்படுத்தும்.$( )-க்குள் parseHTML() வழியே கொடுக்காமல் வெறுமென கொடுத்தோமானால் *All the leaves* எனும் வரி மட்டுமே வெளிப்படும். are falling down எனும் வரி விடுபட்டுவிடும். இவை அனைத்தும் desired changes-ஐ சொடுக்கினால் நடைபெறும்.

Getting an identity is important

Doing the right things at right time

Best things will always comes to you

Big data is the emerging trend

Data Science is different from Big data

You can feel the freedom everywhere
win-win situation is not always possible in family world
Come out of the shell
Female
Male

Desired Changes

:

Desired Changes – :

Doing the right things at right time

Getting an identity is important

All the *leaves* are falling down

Best things will always comes to you

Data Science is different from Big data

You can feel the freedom everywhere

Best things will always comes to you

Data Science is different from Big data

Big data is the emerging trend

Come out of the shell
All the leaves are falling down Desired Changes

17 jQuery □□□□□□□ □□□□□□□□□ □□□□□□□□ □□□□□□□□□□□□□

jQuery கடந்து செல்லும் விதத்தை பின்வரும் இரண்டு நிலைகளில் வரையறுக்கலாம்.

- jQuery Object-ஆக வரையறுக்கப்பட்டுள்ளவற்றுக்குள் கடந்து செல்லல்
- jQuery Object-ஆக வரையறுக்கப்பட்டுள்ளவற்றின் உட்செய்திகளாக இருப்பவற்றுக்குள் கடந்து செல்லல்

இந்த இரண்டிற்கும் each() எனும் method பயன்படுகிறது.

jQuery Object-

:

சாதாரணமாக jQuery object என்றழைக்கப்படும் $("p”) என்பது வலைத்தளப்பக்கத்தில் உள்ள ஒவ்வொரு <p> -ஐயும் கடந்து சென்று நமக்கு வேண்டிய மாற்றங்களை நிகழ்த்தும். அவ்வாறு கடந்து செல்லும்போது நமது விருப்பத்திற்கேற்ப, ஒருசில <p>-ஐத் தவிர்த்து ஒருசில <p>-ஐ மட்டும் மாற்றுவது எப்படி என்று இந்தப் பகுதியில் பார்க்கப் போகிறோம். கீழ்க்கண்ட எடுத்துக்காட்டில் அனைத்து <p>-ன் மீதும் மாற்றங்களை நிகழ்த்தாமல் ஒரு குறிப்பிட்ட நிறத்தில் உள்ள <p>-ஐ மட்டும் சிகப்பு நிறத்தில் மாற்றம் செய்வதற்கான நிரல் கொடுக்கப்பட்டுள்ளது.

```
$("p”).each (function (index,element)
```

{if ($(element).css ("color”) === "rgb(0, 191, 255)”)
{$(element).css ("color”,”rgb(255, 0, 0)”);}
});

$("p”) என்பது jquery-ஐ அனைத்து <p>-ன் வழியே நடத்திச் செல்லும். $("p”).each() {…..} என்பது அந்த இயக்கத்தை வெளிப்படையாகக் குறிப்பிட்டு, ஒவ்வொரு இயக்கத்திலும் சோதனையை நிகழ்த்தும். இதற்குள் ஒரு function() {…..} கொடுக்கப்பட்டுள்ளது. அது index,element எனும் இரண்டு parameters-ஐக் கொண்டுள்ளது. Index-என்பது அடுத்தடுத்த சுழற்சியையும், element என்பது அந்தந்த சுழற்சியில் உள்ள <p> -ஐயும் குறிக்கிறது. இந்த function()-க்குள் வரையறுக்கப்பட்டுள்ள if-ஆனது ஊதா நிறத்தில் உள்ள <p>-ஐ மட்டும் சிவப்பு நிறத்தில் மாற்ற உதவும் சோதனையை நிகழ்த்துகிறது. எனவேதான் ஊதா நிறத்தில் உள்ள "Doing the right things at right time” என்பது சிவப்பு நிறத்தில் மாறியுள்ளது. இவ்வாறாக jQuery object-ஆனது அதன் elements வழியே இயங்கிச் செல்லும் விதத்தை நம்மால் கட்டுப்படுத்தவோ, மாற்றவோ முடியும்.

jQuery Object-

:

சில சமயங்களில் jQuery object-ஆனது Ordered list, Unordered list போன்றவையாக இருப்பின், அதன் கீழ் வரையறுக்கப்பட்டுள்ள விஷயங்களுக்குள் ஒவ்வொன்றாகச் சென்று ஒவ்வொரு விதமான மாற்றங்களை நிகழ்த்துவது எப்படி

என்று இங்கு பார்க்கலாம்.

$("#beverages”).attr ("type”,”circle”) என்பது beverages எனும் id-ஐக் கொண்ட unordered list-ன் கீழ் உள்ள அனைத்தையும் வட்டவடிவத்தை முன்னிறுத்தி வரிசைப்படுத்துகிறது. இதற்கு each() என்பது தேவையில்லை. ஆனால் இவை அனைத்தையும் 1.2.3 எனும் எண்களின் அடிப்படையில் வரிசைப்படுத்தப்பட வேண்டுமெனில் அதற்கு each() என்பது பின்வருமாறு பயன்படும்.

```
$("#beverages”).children().each ( function(i,x)
{$(x).text((i+1)+”.”+$(x).text());} );
```

மேலும் $("li:first-child”) என்பது -ல் முதலாவதாக வரையறுக்கப்பட்டுள்ளவற்றிலும், $("li:first-child”).next() என்பது இரண்டாவதாக வரையறுக்கப்பட்டுள்ளவற்றிலும், $("li:first-child”).siblings() என்பது முதலாவதை விட்டுவிட்டு அதன்கீழ் உள்ள அனைத்திலும், நாம் குறிப்பிட்டுள்ள மாற்றங்களைச் செய்துள்ளது. இவ்வாறாக jQuery object-ன் கீழ் உள்ளவற்றில் நம்மால் மாற்றங்களைச் செய்ய முடியும்.

இவை அனைத்தையும் உள்ளடக்கிய ஒரு எடுத்துக்காட்டு கீழே கொடுக்கப்பட்டுள்ளது.

```
<html>

  <head>
    <script
```

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
    </script>
    <script>
    $(document).ready(function(){
      $("button").click(function(){

        $("#beverages").attr ("type","circle");

        $("#beverages").children().each ( function(i,x)
{$(x).text((i+1)+"."+$(x).text());} );

        $("li:first-child").css ("background-color","skyblue");

        $("li:first-child").next().css ("color","red");

        $("li:first-child").siblings().css ("background-color","lightgreen");

        $("p").each (function (index,element)
          {if ($(element).css ("color") === "rgb(0, 191, 255)")
            {$(element).css ("color","rgb(255, 0, 0)");}
          });
      });
    });
    </script>
  </head>

  <body>
    <ul id='beverages'>
      <li>Coffee</li>
      <li>Tea</li>
```

```
        <li>Milk</li>
    </ul>
    <p style="color:rgb(0, 191, 255)">Doing the right things at right time</p>
    <p>Best things will always comes to you</p>
    <button>Desired Changes</button>
  </body>

</html>
```

view raw03_jquery_traverse.html hosted with by GitHub

:

- Coffee
- Tea
- Milk

Doing the right things at right time

Best things will always comes to you

Desired Changes

Desired Changes- :

- 1.Coffee
- 2.Tea
- 3.Milk

Doing the right things at right time

Best things will always comes to you

Desired Changes



இந்த நூல் JavaScript, JQuery க்கான ஒரு அறிமுகம் மட்டுமே. இவை பற்றி இணையத்தில் பல்வேறு பாடங்களும் காணொளிகளும் கிடைக்கின்றன. அவற்றை தொடர்ந்து படித்து, கற்று, பல்வேறு சாதனைகள் புரிய அனைவருக்கும் வாழ்த்துகள்.

□□□□□□□□□□□□□□□ □□□ □□□□□□□□□□□□

http://freetamilebooks.com/authors/nithyaduraisamy/



[illegible] - Vision

[illegible], [illegible], [illegible]

- Mission

மேற்கண்ட தொலை நோக்கு , பணி இலக்குகளை நோக்கி பயணிக்க, கணியம் குழுவினர், கணியம் அறக்கட்டளையைத் தொடங்கியுள்ளோம். கணியம் இதழில் கட்டுரைகள் வெளியிடுதல், FreeTamilEbooks.com தளத்தில் மின்னூல்கள் வெளியிடுதல், தமிழுக்கான கட்டற்ற மென்பொருட்கள், பிற வளங்கள் உருவாக்குதல், இவற்றுக்கான இணைய வளங்களைப் பேணுதல், விக்கி மூலம் தளத்தில் மின்னூல்களை சீராக்கி வெளியிடுதல், கட்டற்ற மென்பொருட்களுக்கு பரப்புரை நிகழ்ச்சிகள் நடத்துதல் ஆகிய பணிகளை செவ்வனே செய்ய கணியம் அறக்கட்டளை ஆவன செய்யும்.

மேற்கண்ட பணிகளை ஆதரிக்க உங்களை நன்கொடைகளை வேண்டுகிறோம்.
பின்வரும் வங்கிக் கணக்கில் உங்கள் நன்கொடைகளை செலுத்தி,
editor@kaniyam.com க்கு ஒரு மின்னஞ்சல் அனுப்ப வேண்டுகிறோம்.

மிக்க நன்றி !

Kaniyam Foundation
Account Number : 606101010050279
Union Bank Of India
West Tambaram, Chennai
IFSC - UBIN0560618